Date of acceptance Grade

Instructor

# Newton-based Optimization Methods for Noise-Contrastive Estimation

Beenish Qaiser

Helsinki February 15, 2015

M.Sc. Thesis

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

| Tiedekunta — Fakultet — Faculty | Laitos — Institution — Department |
|---|---|
| Faculty of Science | Department of Computer Science |

Tekijä — Författare — Author
Beenish Qaiser

Työn nimi — Arbetets titel — Title

Newton-based Optimization Methods for Noise-Contrastive Estimation

Oppiaine — Läroämne — Subject
Computer Science

| Työn laji — Arbetets art — Level | Aika — Datum — Month and year | Sivumäärä — Sidoantal — Number of pages |
|---|---|---|
| M.Sc. Thesis | February 15, 2015 | 57 pages + 0 appendices |

Tiivistelmä — Referat — Abstract

The main focus of this thesis is on the use of Newton-based optimization methods for the optimization of an objective function that is used in the estimation of unnormalized statistical models. For these models, the probability density function (pdf) is known only up to a multiplicative normalizing factor. A properly normalized pdf is essential in maximum likelihood estimation (MLE) for density estimation. An unnormalized model can be converted into a normalized one by diving it by its integral (or sum) known as the partition function. We can compute the partition function analytically or approximate it using numerical integration methods. Here, we assume that the partition function is not available in a closed form. This makes MLE unsuitable for density estimation. We use a method known as noise-contrastive estimation (NCE) for density estimation of unnormalized models. This method does not rely on numerical integration to approximate the partition function. It estimates the normalizing constant along with the other unknown quantities of the model. The estimation process is based on the optimization of a well-defined objective function also known as the cross-entropy error function. There are no constraints in the optimization and hence, powerful optimization methods designed for unconstrained optimization can be used.

Currently, a first-order optimization method known as the non-linear conjugate gradient (CG) method is being used. However, it has been shown that this method converges at a slow rate in case of large datasets. It is possible to use only a fraction of input samples (data and noise) in order to reduce the computation time of the algorithm. This technique is known as sample average approximation (SAA). However, accuracy of the estimates is compromised when random subsets of input samples are used in order to improve the computational performance of the non-linear CG method. There exists a trade-off between statistical accuracy of the estimates and computational performance of the algorithm. We propose to use the second-order Newton-based optimization methods such as the line search Newton-CG and the trust region Newton-CG methods. These methods produce better search directions than the non-linear CG method as they employ both the gradient and the Hessian of the objective function. However, the Newton method requires the Hessian to be positive definite in order to make progress. Thus, we use the Gauss-Newton approximation to the Hessian to avoid directions of negative curvature in case they occur. Furthermore, every iteration of the Newton method is computationally intensive as it requires computation of the Hessian and its inverse. We integrate the Newton-CG methods with the SAA framework to provide an efficient solution. The gradient is computed using whole sets of input samples whereas the Hessian is computed using random subsets.

As a result, we are able to reduce the computation times of the Newton-CG algorithms without losing the statistical accuracy of the estimates. It is shown that the trust region strategy computationally performs better than the line search strategy. The Newton-CG methods converge faster and do not compromise the accuracy of the estimates even when random subsets consisting of 10% of the input samples only are used during the optimization. This is a considerable improvement over the currently employed non-linear CG method.

Avainsanat — Nyckelord — Keywords
unnormalized models, noise-contrastive estimation, conjugate gradient, line search, trust region

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

# Contents

# 1 Introduction

Optimization plays a central role in machine learning because many algorithms are either implicitly or explicitly optimizing a certain objective function. The performance of the algorithm depends both on the objective function itself and the way it is optimized. This thesis investigates different ways to efficiently optimize an objective function that is used in the estimation of unnormalized statistical models. For these models, the probability density function (pdf) is known only up to a multiplicative normalizing factor, also known as the partition function. For example, the normalizing factor in Markov random fields (MRFs), energy-based models and multi-layer neural networks is often not available in a closed form. A properly normalized pdf is essential in maximum likelihood estimation (MLE) for density estimation [Myu03]. It is, however, possible to estimate unnormalized models using another class of estimation methods that does not rely on the availability of the partition function. It includes methods such as score matching [Hyv05] and noise-contrastive estimation (NCE) [GH12]. In this thesis, we use the latter one for the estimation of unnormalized models. This method is based on the optimization of a well-defined objective function also known as the cross-entropy error function [Bis95]. There are no constraints in the optimization and hence, powerful optimization methods designed for unconstrained optimization can be used. The density estimation problem in noise-contrastive estimation is solved by performing classification between the observed data and some artificially generated noise data. The normalizing constant is considered as an additional parameter and estimated simultaneously along with the other unknown parameters of the model.

Currently, a first-order optimization method known as the non-linear conjugate gradient (CG) method [NW99], [Pol69] is used for optimization of the objective function in noise-contrastive estimation. First-order methods require minimal information to make progress, that is, value and gradient of the objective function are computed iteratively until the algorithm converges. These methods are very simple and easy to implement. They, however, exhibit a linear rate of convergence. For noise-contrastive estimation, it is shown that using more noise samples than data samples gives more and more accurate estimates [GH12]. However, this slows down the optimization process because more and more noise samples need to be processed. Thus, there exists a trade-off between statistical accuracy of the estimates and computational performance of the algorithm. It is possible to use only a fraction of

input samples (data and noise) in order to reduce the computation time of the algorithm. This technique is known as sample average approximation (SAA) [KPH15]. It is shown that when random subsets of input samples are used in optimization, the computational performance of the algorithm improves but the accuracy of the estimates goes down as well [GH12]. The non-linear conjugate gradient method fails to overcome the errors induced by the sample average approximation. Hence, there is room for improvement in the way the optimization is performed.

The main goal of this thesis is to optimize the objective function in noise-contrastive estimation in an iterative fashion which is fast, inexpensive and does not compromise the accuracy of the estimates. The current situation could be improved by utilizing the curvature information (Hessian) of the objective function in addition to the gradient. This leads to second-order optimization methods that use both the gradient and the Hessian to make progress. These methods exhibit a quadratic or super-linear rate of convergence depending upon the chosen parameters [NW99]. In this thesis, we use two different types of optimization methods based on the Newton method [DS96] known as the line search Newton-CG and the trust region Newton-CG methods. The Newton method requires the Hessian to be positive definite at all times in order to make progress. Thus, we use the Gauss-Newton approximation to the Hessian which is always at least positive semi-definite in order to avoid directions of negative curvature in case they occur [Che11]. Furthermore, every iteration of the Newton method is computationally intensive as it requires computation of the Hessian and its inverse. We use the sample average approximation in order to improve the computational performance of the Newton based optimization methods for noise-contrastive estimation. The gradient is computed using whole sets of input samples whereas the Hessian is computed using random subsets of input samples. Following this approach we achieve a reduction in computation times of the Newton-CG algorithms without losing the statistical accuracy of the estimates. This results in a considerable improvement over the currently used non-linear conjugate gradient method.

The thesis is organized as follows: We provide a brief introduction to unnormalized statistical models followed by a discussion of noise-contrastive estimation in Section 2. In Section 3, we go through several topics related to unconstrained optimization. We highlight issues associated with the non-linear conjugate gradient method for optimization of the objective function in noise-contrastive estimation in

Section 4. Our research goals set for this thesis are formally described in Section 5. We discuss the Newton based optimization methods for noise-contrastive estimation in Section 6. We draw a comparison between computational performances of the Newton-CG algorithms with and without using the sample average approximation in Section 7. In the same section, we also compare our results with the ones obtained using the non-linear conjugate gradient method. In Section 8, we briefly go through some of the potential strategies that can be adapted in future to improve the computational performances of the Newton-CG algorithms. Section 9 concludes the thesis.

# 2 Background on Unnormalized Statistical Models

In this section, we provide an overview of unnormalized models and highlight some of the problems associated with conventional techniques for their estimation. We conclude this section with the discussion of noise-contrastive estimation.

## 2.1 Introduction

A parametric statistical model is a family of non-negative functions, each of which is indexed by a finite-dimensional parameter vector $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^d$. We say that a statistical model $\{f_m(\,.\,;\boldsymbol{\theta}) | \, \boldsymbol{\theta} \in \Theta\}$ is normalized if the following normalization condition holds for all $\boldsymbol{\theta} \in \Theta$,

$$\int f_m(\mathbf{u};\boldsymbol{\theta})\,\mathrm{d}\mathbf{u} = 1. \tag{2.1}$$

A statistical model $\{p_m(\,.\,;\boldsymbol{\theta}) | \, \boldsymbol{\theta} \in \Theta\}$ is said to be unnormalized if the condition does not hold. Hence, for normalized models, the $f_m$ are probability density functions while for unnormalized models, the $p_m$ are not. A non-negative function $p_m$ associated with an unnormalized model can always be converted to a pdf by dividing it by its integral (or sum) known as the *partition function*. Given that $p_m(\,.\,;\boldsymbol{\theta})$ is integrable for all $\boldsymbol{\theta}$, the partition function is the integral

$$Z(\boldsymbol{\theta}) = \int p_m(\mathbf{u};\boldsymbol{\theta})\,\mathrm{d}\mathbf{u}. \tag{2.2}$$

The normalized model is specified as

$$f_m(\,.\,;\boldsymbol{\theta}) = \frac{p_m(\,.\,;\boldsymbol{\theta})}{Z(\boldsymbol{\theta})}. \tag{2.3}$$

For example, consider a zero mean uni-variate Gaussian model,

$$p_m(u;\sigma) \propto \exp\left(-\frac{u^2}{2\sigma^2}\right); \qquad \sigma > 0. \tag{2.4}$$

The unnormalized model $p_m(u;\sigma)$ with variance as a parameter can be normalized by dividing it by $Z(\sigma) = \sigma\sqrt{2\pi}$.

It is often the case that the partition function cannot be computed by analytic integration. In this scenario, maximum likelihood estimation (MLE) is no more a viable option for density estimation as reviewed in Section 2.2.1. Approaches to the estimation of unnormalized models can be divided into two categories. The estimation methods which belong to the first category use numerical integration to approximate the partition function. Two broad classes of numerical integration methods are available one of which comprises deterministic integration methods and the other consists of (stochastic) Monte Carlo methods. Deterministic numerical integration becomes computationally very expensive for high dimensional problems. Monte Carlo integration is applicable for larger dimensions but its computational cost is rather high. For detailed information on numerical integration, please refer to [Has61], [GM98] and [Val08]. In the second category, methods such as score matching [Hyv05] and noise-contrastive estimation [GH12] avoid the partition function. For a recent review of score-matching and noise-contrastive estimation, please see [GH13]. Examples of some commonly occurring unnormalized models include Markov networks [KS80], multi-layer neural networks [Bis95] and exponential random graphs [RPKL07]. For a summary on occurrence of unnormalized models, please see Section 3 of [GH13].

## 2.2 Estimation of Unnormalized Models

We use an example from [GH13] to show that the partition function is essential for maximum likelihood estimation. Often, it cannot be computed in a closed form which makes maximum likelihood estimation unsuitable for the estimation of unnormalized models. We then review noise-contrastive estimation which avoids the partition function and estimates the normalizing constant along with the other unknown quantities of the model.

### 2.2.1 Maximum Likelihood Estimation (MLE)

Consider an i.i.d data sample $\mathcal{X} = (x_1, \ldots, x_T)$ drawn from a uni-variate Gaussian distribution with a pdf given as

$$f_m(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left( - \frac{(x-\mu)^2}{2\sigma^2}\right); \qquad \sigma > 0 \text{ and } \mu \in \mathbb{R}. \tag{2.5}$$

We wish to construct estimates of the parameters $\mu, \sigma$ from the observed data $\mathcal{X}$. Since, the observations are drawn independently from $f_m(.\,; \mu, \sigma)$, the joint pdf of $\mathcal{X}$ is given as

$$f_m(\mathcal{X}; \mu, \sigma) = \prod_{t=1}^{T} f_m(x_t; \mu, \sigma). \tag{2.6}$$

The likelihood function is defined by

$$L(\mu, \sigma) \equiv L(\mu, \sigma; \mathcal{X}) = f_m(\mathcal{X}; \mu, \sigma) \tag{2.7}$$

where $\mathcal{X}$ is fixed and $\mu$, $\sigma$ are allowed to vary. The likelihood function is a function of parameters of a statistical model and not a pdf. It is only defined up to a constant of proportionality. In maximum likelihood estimation, the parameters of the model are often estimated by optimizing $\log L$, the log-likelihood [Myu03]. The log-likelihood of the uni-variate Gaussian model in (2.5) is given as

$$\ell(\mu, \sigma) = -\, N \ln(\sigma\sqrt{2\pi}) - \frac{1}{2\sigma^2} \sum_{t=1}^{T} (x_t - \mu)^2. \tag{2.8}$$

The maximum likelihood estimates of $\mu$ and $\sigma$ are

$$\widehat{\mu} = \frac{\sum_{t=1}^{T} x_t}{T}; \quad \text{and} \quad \widehat{\sigma}^2 = \frac{\sum_{t=1}^{T} (x_t - \widehat{\mu})}{T}. \tag{2.9}$$

Now consider an unnormalized Gaussian model given as

$$p_m(x; \mu = 0,\, \sigma) \propto \exp\left( - \frac{x^2}{2\sigma^2}\right); \qquad \sigma > 0. \tag{2.10}$$

In analogy to the log-likelihood, we can consider $\widetilde{\ell}$,

$$\widetilde{\ell}(\sigma) = -\, \frac{1}{2\sigma^2} \sum_{t=1}^{T} x_t^2. \tag{2.11}$$

As the variance is positive, $\widetilde{\ell}(\sigma)$ can be minimized by making $\sigma$ as large as possible. This estimate is obtained irrespective of the data and is not meaningful.

### 2.2.2 Noise-Contrastive Estimation (NCE)

Consider an i.i.d data sample $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{T_d})$ of a random variable $\mathbf{x} \in \mathbb{R}^n$ with an unknown pdf $f_d$. The data-pdf $f_d$ is modelled by means of a parametrized family of non-negative functions. It is assumed that the data-pdf $f_d$ is the same as the model-function $p_m$ for some parameter vector $\boldsymbol{\theta}^*$, that is, $f_d(.) = p_m(. \,; \boldsymbol{\theta}^*)$. The basic principle underlying noise-contrastive estimation is the construction of an estimate $\widehat{\boldsymbol{\theta}}$ from $\mathcal{X}$ by comparing it with some reference data $\mathcal{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_{T_n})$ of a random variable $\mathbf{y} \in \mathbb{R}^n$ with a pdf $f_n$ whose properties are known. If the reference distribution $f_n$ is known, it is obvious that $f_d$ can be obtained from the ratio $f_d/f_n$ [GH12].

Let $\boldsymbol{\gamma}$ denote the unknown parameters of an unnormalized statistical model, say $p\,(. \,; \boldsymbol{\gamma})$ and $Z(\boldsymbol{\gamma})$ is the partition function. In noise-contrastive estimation, the partition function is avoided by replacing it with a scaling parameter, say $c$. The model that we now estimate is

$$\ln p_m(. \,; \boldsymbol{\theta}) = \ln p\,(. \,; \boldsymbol{\gamma}) + c, \tag{2.12}$$

where $\boldsymbol{\theta} = (\boldsymbol{\gamma}, c)$. The model-function $p_m$ is not normalized for all $(\boldsymbol{\gamma}, c)$ but only for a specific value of $c$ so that the condition in (2.1) is satisfied. The estimated value $\widehat{c}$ provides an estimate of $\ln \dfrac{1}{Z(\widehat{\boldsymbol{\gamma}})}$. The density estimation problem in noise-contrastive estimation is solved by performing classification between the two data sets $\mathcal{X}$ and $\mathcal{Y}$, which leads to an optimization problem involving the well-known cross-entropy error function [Bis95] as its objective function.

Let $\mathcal{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{T_d+T_n})$ denote union of the two data sets $\mathcal{X}$ and $\mathcal{Y}$. Each data point $\mathbf{u}_t$ is assigned a binary class label $C_t : C_t = 1$ if $\mathbf{u}_t \in \mathcal{X}$ and $C_t = 0$ if $\mathbf{u}_t \in \mathcal{Y}$. The log-ratio between the two functions $p_m$ and $f_n$, given by

$$G(\mathbf{u}; \boldsymbol{\theta}) = \ln p_m(\mathbf{u}; \boldsymbol{\theta}) - \ln f_n(\mathbf{u}) \tag{2.13}$$

discriminates between the two datasets. The resulting value is used by a logistic activation function to produce the posterior probability that $C_t = 1$ and is given as

$$h(\mathbf{u}; \boldsymbol{\theta}) = \frac{1}{1 + \nu \exp\big(-G(\mathbf{u}; \boldsymbol{\theta})\big)}, \tag{2.14}$$

where $\nu$ is defined as the ratio $T_n/T_d$. The conditional log-likelihood of $\boldsymbol{\theta}$ given the data set $\mathcal{U}$ assuming that the class labels $C_t$ are Bernoulli distributed and independent is given as

$$\ell(\boldsymbol{\theta}) = \sum_{t=1}^{T=T_d+T_n} C_t \ln h(\mathbf{u}_t;\boldsymbol{\theta}) + (1-C_t)\ln\big(1-h(\mathbf{u}_t;\boldsymbol{\theta})\big). \tag{2.15}$$

The negative log-likelihood $-\ell(\boldsymbol{\theta})$ is also known as the *cross-entropy error function* [Bis95]. The estimate $\widehat{\boldsymbol{\theta}}_T$ is defined to be the argument which minimizes the following function with respect to $\boldsymbol{\theta}$,

$$J_T(\boldsymbol{\theta}) = \frac{1}{T_d}\left\{ \sum_{t=1}^{T=T_d+T_n} - C_t \ln h(\mathbf{u}_t;\boldsymbol{\theta}) - (1-C_t)\ln\big(1-h(\mathbf{u}_t;\boldsymbol{\theta})\big) \right\}. \tag{2.16}$$

The estimate $\widehat{\boldsymbol{\theta}}_T$ is such that $p_m(.\,;\widehat{\boldsymbol{\gamma}})$ matches shape of $f_d$ and $\widehat{c}$ provides proper scaling. It converges in probability to $\boldsymbol{\theta}^*$. For detailed information on noise-contrastive estimation, please refer to [GH12].

# 3 Background on Unconstrained Optimization

In this section, we provide a brief introduction to core concepts and basic methods used in unconstrained optimization. For a comprehensive literature review of unconstrained optimization, please refer to Dennis & Schnabel [DS96], Griva, Nash & Sofer [GNS08] and Nocedal & Wright [NW99].

## 3.1 Optimization Problem

Consider an optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad F(\mathbf{x}), \tag{3.1}$$

where $F : \mathbb{R}^n \to \mathbb{R}$ is a smooth function and $\mathbf{x} \in \mathbb{R}^n$ is a real vector with $n \geq 1$ components. The objective function $F$ is minimized with no restrictions on the values of these $n$ real variables. This is known as *unconstrained optimization*. A solution to the minimization problem in (3.1) is a point, say $\mathbf{x}^* \in \mathbb{R}^n$, at which the function $F$ attains its minimum value. Furthermore,

- A point $\mathbf{x}^*$ is a *global minimizer,* if $F$ attains its least value in $\mathbb{R}^n$ at $\mathbf{x}^*$, that is, $F(\mathbf{x}^*) \leq F(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n$. A point $\mathbf{x}^*$ is a *strict global minimizer*, if $F(\mathbf{x}^*) < F(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{x} \neq \mathbf{x}^*$.
- A point $\mathbf{x}^*$ is a *local minimizer*, if there exists a neighbourhood $\mathcal{N} = \left\{\mathbf{x} \in \mathbb{R}^n \,\middle|\, \|\mathbf{x} - \mathbf{x}^*\| < \lambda\right\}$ such that $F$ attains its least value in $\mathcal{N}$ at $\mathbf{x}^*$, that is, $F(\mathbf{x}^*) \leq F(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{N}$. A point $\mathbf{x}^*$ is a *strict local minimizer*, if $F(\mathbf{x}^*) < F(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{N}$ and $\mathbf{x} \neq \mathbf{x}^*$.

## 3.2 Optimality Conditions for Optimization

As we do not know the global structure of $F$ over $\mathbb{R}^n$, it is very difficult to locate its global minimum. The only way to predict the overall structure of $F$ is to visit every point in $\mathbb{R}^n$. This, however, dramatically increases the computational cost of an optimization algorithm as several function values and first-order derivatives (gradient) of $F$ are required to make progress. Some algorithms may also need to compute second-order derivatives (Hessian) of $F$. Thus, most algorithms are only able to locate a local minimum. To check whether a point is a local minimizer or not we still need to compare it with every other point in its immediate neighbourhood. When the function $F$ is smooth, there are other practical ways to recognize a local minimum. Given that the function $F$ is twice continuously differentiable, we can identify a local minimizer $\mathbf{x}^*$ by examining just the gradient $\nabla F(\mathbf{x}^*)$ and the Hessian $\nabla^2 F(\mathbf{x}^*)$. Below is a set of optimality conditions that must be satisfied by a local minimizer:

- **First-Order Necessary Conditions** (Theorem 2.2 [NW99]): If $\mathbf{x}^*$ is a local minimizer of a continuous and differentiable function $F : \mathbb{R}^n \to \mathbb{R}$, then the gradient $\nabla F(\mathbf{x}^*) = 0$.
- **Second-Order Necessary Conditions** (Theorem 2.3 [NW99]): If $\mathbf{x}^*$ is a local minimizer of a continuous and twice differentiable function $F : \mathbb{R}^n \to \mathbb{R}$ and the gradient $\nabla F(\mathbf{x}^*) = 0$, then the Hessian $\nabla^2 F(\mathbf{x}^*)$ is positive semi-definite.
- **Second-Order Sufficient Conditions** (Theorem 2.4 [NW99]): Following the first and second-order necessary conditions if the gradient $\nabla F(\mathbf{x}^*) = 0$ and the Hessian $\nabla^2 F(\mathbf{x}^*)$ is positive definite, then $\mathbf{x}^*$ is a strict local minimizer of $F$.

## 3.3 Optimization Strategies

An algorithm for the minimization problem in (3.1) is implemented in form of an iterative procedure. Given an initial guess, say $\mathbf{x}_0$, a sequence $\{\mathbf{x}_k\}$ is generated by invoking the update rule,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \tag{3.2}$$

where $\mathbf{p}_k \in \mathbb{R}^n$ is known as the *search direction* and $\alpha_k$ specifies the length of the move along $\mathbf{p}_k$ known as the *step size*. A good search direction should reduce the objective function, that is, at any point $F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$. We call such a search direction $\mathbf{p}_k$ a *descent direction* and it satisfies the following requirement

$$\nabla F(\mathbf{x}_k)^T \mathbf{p}_k < 0. \tag{3.3}$$

To see why, consider the *directional derivative* which is the rate of change of $F$ at $\mathbf{x}_k$ in the direction of $\mathbf{p}_k$ given as

$$\nabla F(\mathbf{x}_k) \,.\, \mathbf{p}_k = \|\nabla F(\mathbf{x}_k)\| \, \| \mathbf{p}_k\| \cos(\vartheta_k), \tag{3.4}$$

where $\vartheta_k$ is the angle between $\mathbf{p}_k$ and $\nabla F(\mathbf{x}_k)$. The rate at which a function value changes depends on the value of the cosine function. If $\vartheta_k = 90^\circ$ then steps along $\mathbf{p}_k$ do not reduce the value of $F$. If $0 \leq \vartheta_k < 90^\circ$, the value of $F$ increases. We are looking for a search direction that makes an angle greater than $90^\circ$ with $\nabla F(\mathbf{x}_k)$. In this case, value of the cosine function lies in the interval $-1 \leq \cos(\vartheta_k) < 0$ leading to $\nabla F(\mathbf{x}_k)^T \mathbf{p}_k < 0$. Furthermore, $\mathbf{p}_k$ is a *direction of negative curvature*, if

$$\mathbf{p}_k^T \nabla^2 F(\mathbf{x}_k) \, \mathbf{p}_k < 0. \tag{3.5}$$

The algorithm is terminated either when a solution $\mathbf{x}^*$ has been approximated with sufficient accuracy. In practice, we run the algorithm until the Euclidean norm of $\nabla F(\mathbf{x}_k)$ is sufficiently close to zero specified by a scalar, say $\epsilon$. A general optimization setup is as follows: At every $k^{th}$ iteration, the algorithm

- computes a search direction $\mathbf{p}_k$ such that $\nabla F(\mathbf{x}_k)^T \mathbf{p}_k < 0$.
- computes an optimal value for the step size $\alpha_k > 0$ such that $F(\mathbf{x}_k + \alpha_k \mathbf{p}_k) < F(\mathbf{x}_k)$.

There are two fundamental strategies for moving from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$ known as *line search* and *trust region*. These two strategies differ by the order in which they compute the search direction and the step size. The line search methods first

calculate the search direction and then find an optimal value of the step size whereas, the trust region methods calculate the search direction and the step size simultaneously by forming a region around the current iterate which restricts the norm of the search direction to be no greater than the radius of the region.

### 3.3.1 Optimization Strategy I: Line Search

A line search method first computes a search direction $\mathbf{p}_k$ and then decides how far to move along $\mathbf{p}_k$ specified by a step size $\alpha_k$. If $\alpha_k$ is too small, then the algorithm will converge very slowly. On the other hand, if $\alpha_k$ is not chosen small enough, then the algorithm may fail to achieve a desired reduction in the function value. In fact, a search is carried out along the line $\mathbf{x}_k + \alpha\mathbf{p}_k$ to generate an optimal value for $\alpha_k \in \mathbb{R}_{>0}$ which minimizes the predicted function value. Hence, the step size $\alpha_k$ is obtained as a solution to the following sub-problem,

$$\min_{\alpha \in \mathbb{R}_{>0}} \quad F(\mathbf{x}_k + \alpha\mathbf{p}_k). \tag{3.6}$$

The cost of this sub-problem is expected to be much less than that of the original problem in (3.1). We can derive an exact solution of (3.6) if the objective function is of a favourable form. Otherwise, we settle for an inexact solution.

#### 3.3.1.1 Exact Line Search for Convex Quadratic Functions

Consider a convex quadratic objective function,

$$\phi(\mathbf{x}) = \frac{1}{2}\,\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}, \tag{3.7}$$

where $\mathbf{x} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a symmetric and positive-definite matrix. The gradient of $\phi$ is given as,

$$\nabla\phi(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}. \tag{3.8}$$

If a line search method is used to minimize $\phi$, then the step size $\alpha_k$ is a one-dimensional minimizer of the function,

$$\phi(\mathbf{x}_k + \alpha\mathbf{p}_k) = \frac{1}{2}\,(\mathbf{x}_k + \alpha\mathbf{p}_k)^T\mathbf{A}(\mathbf{x}_k + \alpha\mathbf{p}_k) - \mathbf{b}^T(\mathbf{x}_k + \alpha\mathbf{p}_k). \tag{3.9}$$

We set $\dfrac{\partial\phi(\mathbf{x}_k + \alpha\mathbf{p}_k)}{\partial\alpha} = 0$, to obtain an exact value for the step size $\alpha_k$ given as,

$$\boxed{\alpha_k = -\frac{\nabla\phi(\mathbf{x}_k)^T\mathbf{p}_k}{\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k}.} \tag{3.10}$$

#### 3.3.1.2 Inexact Line Search

For general non-linear objective functions, it may not be possible to compute an exact solution for $\alpha_k$ in a closed form or perhaps the cost of obtaining the solution is too high. In such situations, we seek an inexact solution to (3.6). Inexact line search for $\alpha_k$ is carried out in the following three phases:

- **Initial Phase:** Choose an initial step $\alpha_0$.
- **Bracketing Phase:** Find an interval containing acceptable step lengths $\{a, b\}$.
- **Selection Phase:** Candidate values for $\alpha_k$ are obtained by shrinking the set $\{a, b\}$ through interpolation and extrapolation of known function values and its derivatives. The search is complete upon the fulfilment of a termination criteria.

Thorough understanding of the step size selection procedure requires an elaborate discussion of its own. Here, we only discuss a set of termination conditions used by the inexact line search algorithms which guarantees that an optimal value for $\alpha_k$ is located upon termination. For further information, please refer to Section 3.4 of [NW99].

It seems acceptable to require that $F(\mathbf{x}_k + \alpha_k \mathbf{p}_k) < F(\mathbf{x}_k)$. However, this simple condition does not guarantee that the sequence $\{\mathbf{x}_k\}$ will converge to $\mathbf{x}^*$ [DS96]. Additionally, it is required that the average rate of decrease from $F(\mathbf{x}_k)$ to $F(\mathbf{x}_{k+1})$ is at least some prescribed fraction of the initial rate of decrease in the direction of $\mathbf{p}_k$. The initial rate of decrease is given by the directional derivative of $F(\mathbf{x}_k)$ in the direction $\mathbf{p}_k$, that is, $\nabla F(\mathbf{x}_k)^T \mathbf{p}_k$. Hence, we choose $\alpha_k > 0$ from a set of values of $\alpha$ which satisfies the inequality

$$F(\mathbf{x}_k + \alpha \mathbf{p}_k) \leq F(\mathbf{x}_k) + c_1 \alpha \nabla F(\mathbf{x}_k)^T \mathbf{p}_k, \tag{3.11}$$

for some constant $c_1 \in (0, 1)$. We refer to this as the *sufficient decrease condition*. The right hand side of (3.11) is a linear function of $\alpha$, say $l(\alpha)$. Since $\mathbf{p}_k$ is a descent direction and $\nabla F(\mathbf{x}_k)^T \mathbf{p}_k < 0$, it is implied that the line $l(\alpha)$ has a negative slope. The left hand side of (3.11) denotes the predicted function value for a given $\alpha$. The graph of $F(\mathbf{x}_k + \alpha \mathbf{p}_k)$ may turn either upward or downward as $\alpha$ increases from 0. The constant $c_1$ is used to guarantee that the graph of $l(\alpha)$ lies above that of $F(\mathbf{x}_k + \alpha \mathbf{p}_k)$ for smaller values of $\alpha$. Intervals containing acceptable values of $\alpha$ are indicated in Figure 1.

The sufficient decrease condition ensures decrease in the function value when we move from $F(\mathbf{x}_k)$ to $F(\mathbf{x}_{k+1})$ but it is possible that the chosen step size is too small to make any reasonable progress. For reasonably large steps, it is required that the rate of decrease of $F(\mathbf{x}_{k+1})$ in the direction $\mathbf{p}_k$ is larger than some prescribed fraction of the initial rate of decrease. Hence, we choose $\alpha_k > 0$ from a set of values of $\alpha$ which satisfies the second inequality

$$\nabla F(\mathbf{x}_k + \alpha \mathbf{p}_k)^T \mathbf{p}_k \geq c_2 \nabla F(\mathbf{x}_k)^T \mathbf{p}_k, \tag{3.12}$$

for some constants $0 < c_1 < c_2 < 1$, where $c_1$ is the constant from (3.11). This is known as the *curvature condition*. It ensures that the predicted slope $\nabla F(\mathbf{x}_k + \alpha \mathbf{p}_k)^T \mathbf{p}_k$ is greater than $c_2$ times the initial slope $\nabla F(\mathbf{x}_k)^T \mathbf{p}_k$. We are looking for an iterate $\mathbf{x}_{k+1}$ along $\mathbf{p}_k$ where the gradient is less negative than it is at the current iterate $\mathbf{x}_k$. We terminate the search as soon as the gradient becomes slightly less negative than before or even positive. Intervals containing acceptable values of $\alpha$ are indicated in Figure 2.

In 1969, Philip Wolfe put together the sufficient decrease condition and the curvature condition to form a set of termination conditions for inexact line search known as the *Wolfe conditions* [Wol69], [Wol71]. The curvature condition in Wolfe conditions is slightly modified to guarantee that $\alpha_k$ at least lies in a broad neighbourhood of the minimizers of $F(\mathbf{x}_k + \alpha \mathbf{p}_k)$ forming another set of termination conditions known as the *strong Wolfe conditions*, given as,

$$F(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \leq F(\mathbf{x}_k) + c_1 \alpha_k \nabla F(\mathbf{x}_k)^T \mathbf{p}_k, \tag{3.13}$$

$$|\nabla F(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k| \leq c_2 |\nabla F(\mathbf{x}_k)^T \mathbf{p}_k|. \tag{3.14}$$

For practical implementation of the above termination criteria, please see Algorithms 3.2 and 3.3 described in [NW99].

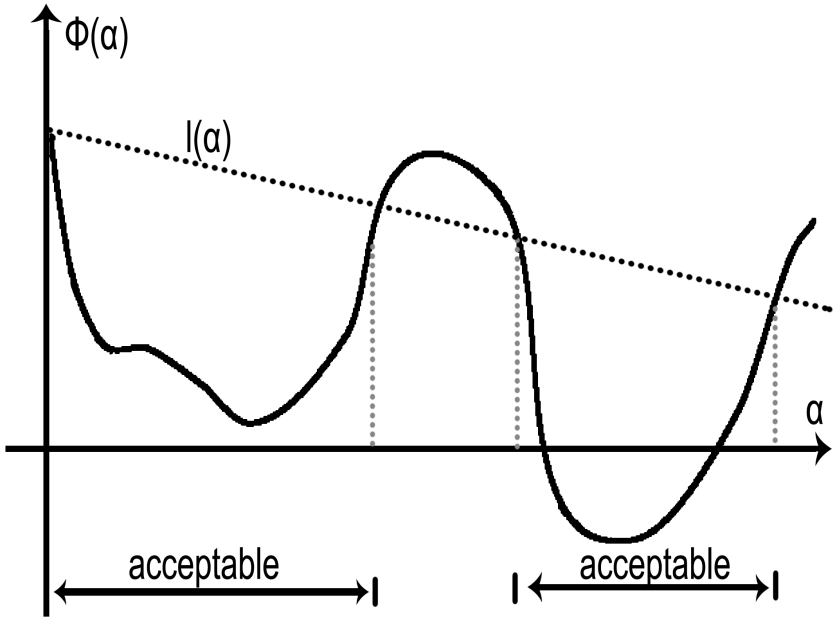



Figure 1: Sufficient Decrease Condition (Figure 3.3 [NW99])

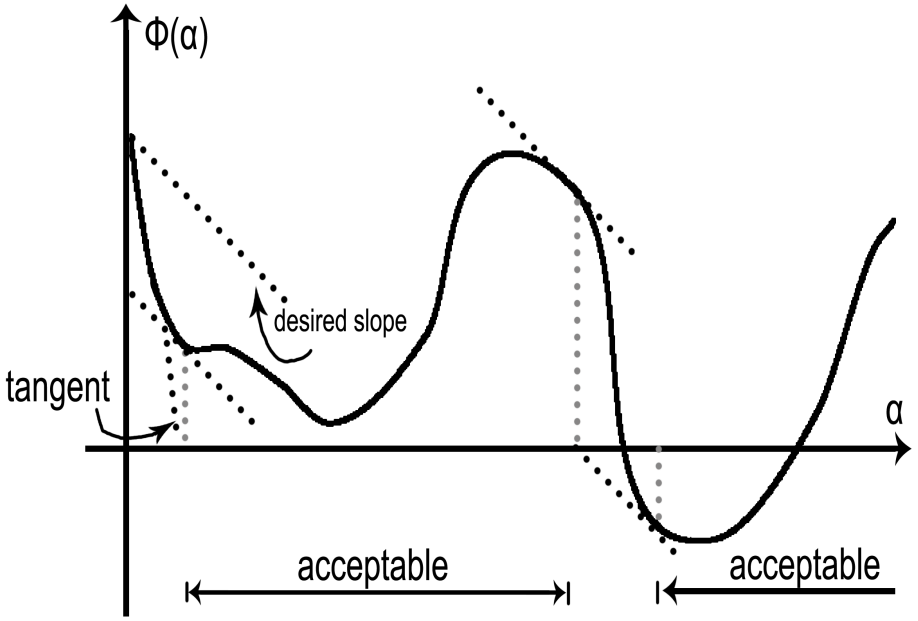



Figure 2: Curvature Condition (Figure 3.4 [NW99])

### 3.3.2 Optimization Strategy II: Trust Region

In trust region methods, the function $F$ is approximated by a quadratic function of $\mathbf{p}$, say $m(\mathbf{p})$, using second-order Taylor series expansion centred at the current iterate $\mathbf{x}_k$ and is given as

$$F(\mathbf{x}_k + \mathbf{p}) \approx m(\mathbf{p}) = F(\mathbf{x}_k) + \nabla F(\mathbf{x}_k)^T \mathbf{p} + \frac{1}{2}\,\mathbf{p}^T \mathbf{H}_k\,\mathbf{p}, \tag{3.15}$$

where $\mathbf{H}_k$ is either the exact Hessian $\nabla^2 F(\mathbf{x}_k)$ or an approximation matrix. The quadratic model $m(.)$ is believed to provide an adequate representation of $F$ within a certain region known as the trust region and the norm of the search direction $\mathbf{p}_k$ is constrained to be no greater than the trust region radius, say $\Delta_k$. Hence, at every $k^{th}$ iteration of a trust region method, $\mathbf{p}_k$ is obtained as a solution to the following constrained sub-problem:

$$\min_{\mathbf{p}\in\mathbb{R}^n} \quad F(\mathbf{x}_k) + \nabla F(\mathbf{x}_k)^T \mathbf{p} + \frac{1}{2}\,\mathbf{p}^T \mathbf{H}_k\,\mathbf{p} \qquad \text{s.t.} \parallel \mathbf{p}\parallel \leq \Delta_k. \tag{3.16}$$

Trust regions with large radius are prone to inaccurate approximations in regions farther away from the current iterate. It is also a possibility that the minimum of the model $m(.)$ is located far away from the minimum of $F$. On the other hand, trust regions with small radius fail to encapsulate any informative regions. This causes the algorithm to make very small progress during each iteration.

The value of the trust region radius $\Delta_k$ is directly related to the iterative performance of the algorithm. If at the $k^{th}$ iteration the model $m(.)$ provides a reasonable prediction of the function $F$, then the size of the trust region is increased. Otherwise, the trust region radius is reduced for the next iteration. This decision is based on an agreement between the quadratic model and the objective function which we define in form of a ratio, say $\rho_k$, and is given as

$$\rho_k = \frac{F(\mathbf{x}_k) - F(\mathbf{x}_k + \mathbf{p}_k)}{m(0) - m(\mathbf{p}_k)}, \tag{3.17}$$

where the numerator is the actual reduction and the denominator is the predicted reduction which will always be non-negative. Hence, if $\rho_k$ is negative it means that $F(\mathbf{x}_k + \mathbf{p}_k) > F(\mathbf{x}_k)$ and therefore, the step is rejected instantly, the trust region shrinks for the next iteration and no progress is made. For more information on trust region methods, please see Chapter 4 of [NW99]. The following algorithm describes the process of adjusting the trust region radius.

**Algorithm** *TrustRegion*
**Input:** $\mathbf{x}_0$, $\bar{\Delta} > 0$, $\Delta_0 \in (0, \bar{\Delta})$, $\eta \in [0, \frac{1}{4})$ and $\epsilon$

**Output:** $\mathbf{x}^*$

1. Set $k = 0$
2. Evaluate $\nabla F(\mathbf{x}_k)$
3. **while** (the Euclidean norm of $\nabla F(\mathbf{x}_k)$ is not sufficiently close to zero specified by the scalar $\epsilon$)
4. **do** Obtain $\mathbf{p}_k$ by solving (3.16)
5. Set $\rho_k = \dfrac{F(\mathbf{x}_k) - F(\mathbf{x}_k + \mathbf{p}_k)}{m(0) - m(\mathbf{p}_k)}$
6. **if** $\left(\rho_k < 1/4\right)$
7. $\Delta_{k+1} = \dfrac{1}{4} \|\mathbf{p}_k\|$
8. **else**
9. **if** $\left(\rho_k > 3/4 \quad \text{and } \|\mathbf{p}_k\| = \Delta_k\right)$
10. $\Delta_{k+1} = \min\left(2\Delta_k, \bar{\Delta}\right)$
11. **else**
12. $\Delta_{k+1} = \Delta_k$
13. end(**if**)
14. end(**if**)
15. **if** $(\rho_k > \eta)$
16. $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$
17. **else**
18. $\mathbf{x}_{k+1} = \mathbf{x}_k$
19. end(**if**)
20. $k = k + 1$
21. Evaluate $\nabla F(\mathbf{x}_k)$
22. end(**while**)
23. **return** $\mathbf{x}^* = \mathbf{x}_k$

## 3.4 Steepest Descent Method

The steepest descent method was first proposed by Cauchy in 1847 [Cau47]. It is the simplest gradient method for unconstrained optimization. A gradient is a vector that points in the direction of greatest increase of a function. It is a natural choice to move in the direction opposite to the gradient when minimum of a function is sought. The search direction generated by this method is of the form

$$\mathbf{p}_k = -\nabla F(\mathbf{x}_k), \tag{3.18}$$

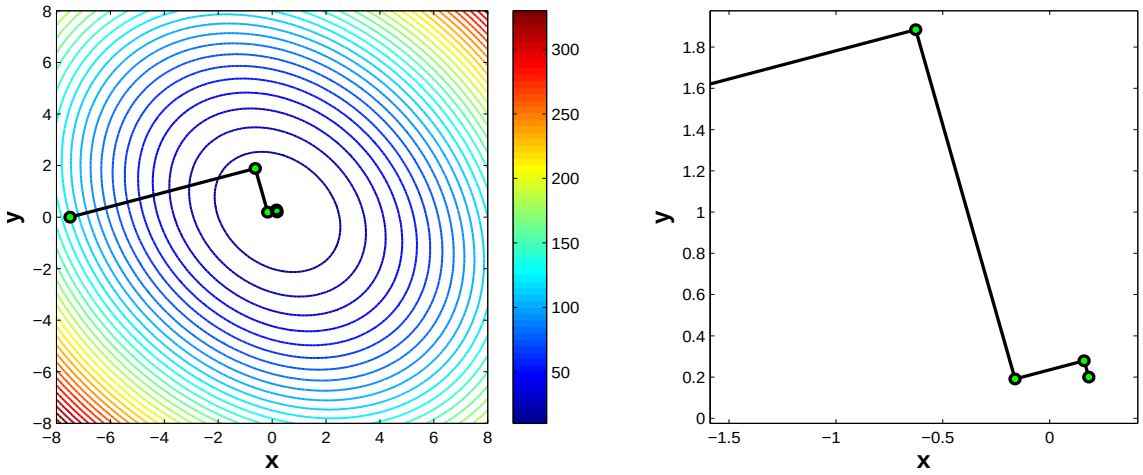
which we call the steepest descent direction. The step size $\alpha_k > 0$ is either an exact or inexact solution to the following line search problem

$$\min_{\alpha \in \mathbb{R}_{>0}} \quad F(\mathbf{x}_k + \alpha \mathbf{p}_k). \tag{3.19}$$

The steepest descent methods move from the current iterate $\mathbf{x}_k$ to the next iterate $\mathbf{x}_{k+1}$ using the update rule

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla F(\mathbf{x}_k). \tag{3.20}$$

The next iterate $\mathbf{x}_{k+1}$ follows the direction of the negated gradient which may result in a zig-zag pattern as shown in Figure 3.




(a) (b)

Figure 3: Minimization of the quadratic function $\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$; $\mathbf{x} \in \mathbb{R}^2$ using the steepest descent method. The algorithm converges after $k = 4$ iterations. Note the zig-zag path in (b).

Steepest descent methods are globally convergent. In case of convex quadratic functions, these methods are shown to be super-linearly convergent in two-dimensional space [BB88]. For general non-linear functions, steepest descent methods exhibit a linear rate of convergence given as,

$$\frac{F(\mathbf{x}_{k+1}) - F(\mathbf{x}^*)}{F(\mathbf{x}_k) - F(\mathbf{x}^*)} \leq \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)^2, \tag{3.21}$$

where $\lambda_1 \leq \ldots \leq \lambda_n$ are the eigenvalues of $F$ at $\mathbf{x}^*$. These methods can be very slow if the ratio $\lambda_n/\lambda_1$ is very large. If the eigenvalues of the Hessian at $\mathbf{x}^*$ differ by several orders of magnitude, a steepest descent algorithm could end up spending a very large number of iterations before locating a solution. It is shown that the number of steepest descent iterations using either exact or inexact line searches required to find an iterate at which the norm of the function's gradient is less than a prescribed error term $\epsilon$ is essentially a multiple of $1/\epsilon^2$ [CGT12].

## 3.5 Newton Method

Newton method [DS96] models the objective function as a quadratic function of $\mathbf{p}$ using second-order Taylor series expansion centred at the current iterate $\mathbf{x}_k$ as follows

$$F(\mathbf{x}_k + \mathbf{p}) \approx m_k(\mathbf{p}) = F(\mathbf{x}_k) + \nabla F(\mathbf{x}_k)^T \mathbf{p} + \frac{1}{2}\,\mathbf{p}^T \mathbf{H}_k \mathbf{p}, \tag{3.22}$$

where $\mathbf{H}_k$ is either the exact Hessian $\nabla^2 F(\mathbf{x}_k)$ or an approximation matrix. The search direction is obtained by solving $\nabla m_k(\mathbf{p}) = 0$,

$$\mathbf{H}_k\,\mathbf{p}_k = -\nabla F(\mathbf{x}_k), \tag{3.23}$$

$$\mathbf{p}_k = -\big[\mathbf{H}_k\big]^{-1} \nabla F(\mathbf{x}_k). \tag{3.24}$$

$\mathbf{p}_k$ is known as the Newton direction, or the Newton step. Transition from the current iterate $\mathbf{x}_k$ to the next iterate $\mathbf{x}_{k+1}$ is carried out using the update rule

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \big[\mathbf{H}_k\big]^{-1} \nabla F(\mathbf{x}_k). \tag{3.25}$$

The Newton method always uses unit step sizes to move from one iterate to another. In case of convex quadratic objective functions, the Newton method outperforms the steepest descent method by locating the minimum in just one step as shown in Figure 4. Given that the starting point $\mathbf{x}_0$ is sufficiently close to $\mathbf{x}^*$ and a unit step size is used in every iteration, the method has well-defined convergence properties.

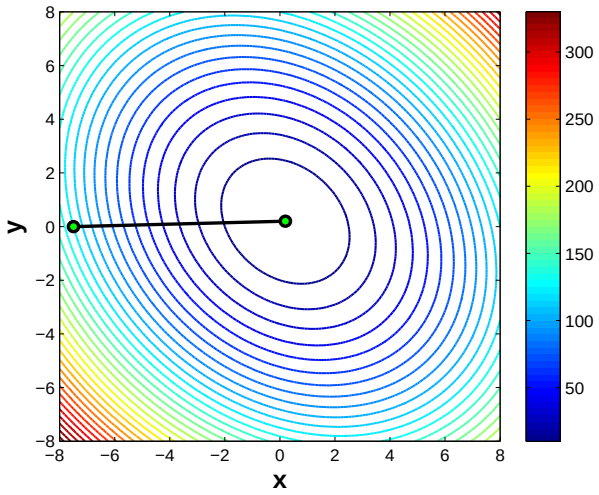



Figure 4: Minimization of the quadratic function $\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$; $\mathbf{x} \in \mathbb{R}^2$ using the Newton method. It locates the minimum of the function in just one iteration.

The rate of convergence of the sequence $\{\mathbf{x}_k\}$ is quadratic, and the sequence of gradient norms converges $\| \nabla F(\mathbf{x})\|$ quadratically to zero (Theorem 5.2.1 [DS96]). In general, the unit step step sizes do not guarantee that the function value is decreasing with every iteration. This causes the iterates of the Newton method to become equally attracted to the minimum or maximum of the objective function. Indeed, the method is just trying to solve $\nabla F(\mathbf{x}) = 0$. Moreover, the Newton method may not necessarily converge if the starting point is located far away from the solution. We discuss this in detail in Section 6 when we present the Newton based optimization methods used in this thesis to achieve our desired goals.

## 3.6 Conjugate Gradient (CG) Method

The conjugate gradient method was originally proposed by Hestenes and Stiefel in 1952 [HS52] to solve systems of linear equations of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{3.26}$$

where $\mathbf{x} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a symmetric and positive-definite matrix. The objective function associated with this linear system is the convex quadratic function of Section 3.3.1.1 and is given as

$$\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}. \tag{3.27}$$

Note that the solution $\mathbf{x}^*$ of the linear system in (3.26) is equivalent to the minimizer of $\phi$. It is obtained by setting the gradient of $\phi$,

$$\nabla\phi(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}, \tag{3.28}$$

to zero. The gradient is the residual $\mathbf{r}$ of the linear system. Given an initial guess $\mathbf{x}_0$, conjugate gradient methods use the update rule,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \tag{3.29}$$

to generate a sequence of iterates $\{\mathbf{x}_k\}$ that converges to $\mathbf{x}^*$. The step size $\alpha_k$ is derived from (3.10) given as

$$\alpha_k = -\frac{\mathbf{r}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}. \tag{3.30}$$

The search directions determined by the conjugate gradient method are said to be conjugate to the matrix $\mathbf{A}$, that is,

$$\mathbf{p}_i^T \mathbf{A} \mathbf{p}_j = 0, \qquad \forall\, i \neq j. \tag{3.31}$$

The set of $n$ such conjugate search directions is linearly independent and hence spans the whole space $\mathbb{R}^n$. Conjugate search directions are important in a way that the function $\phi$ is minimized in $n$ steps by successively minimizing it along each of the conjugate search directions (Theorem 5.1 [NW99]), see Figure 5.

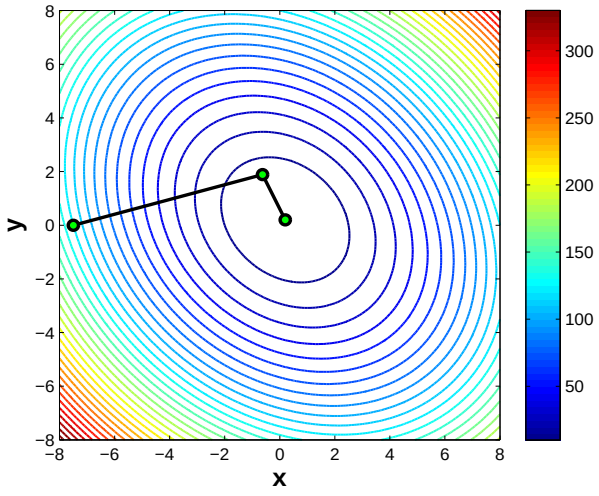



Figure 5: Minimization of the quadratic function $\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$; $\mathbf{x} \in \mathbb{R}^2$ using the conjugate gradient method. The algorithm converges in $k = 2$ iterations.

The residuals $\{\mathbf{r}_k\}$ are mutually orthogonal (Theorem 5.3 [NW99]), and an iteration formula for the residuals is derived from the relations (3.28) and (3.29):

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k \mathbf{A}\mathbf{p}_k. \tag{3.32}$$

The basic idea for constructing a set of conjugate search directions comes from *Gram-Schimdt orthogonalization* [Lay12]. Conjugate gradient methods use the steepest descent direction as the initial search direction, that is, $\mathbf{p}_0 = -\mathbf{r}_0$. We can compute $\mathbf{p}_{k+1}$ using the iteration formula,

$$\mathbf{p}_{k+1} = -\mathbf{r}_{k+1} + \beta_{k+1}\mathbf{p}_k, \tag{3.33}$$

where the coefficient $\beta_{k+1}$ is of the form

$$\beta_{k+1} = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}. \tag{3.34}$$

An algorithm for the conjugate gradient method can formally be stated as follows:

**Algorithm** *ConjugateGradient*
**Input:** $\mathbf{x}_0$, $\mathbf{A}$, and $\mathbf{b}$

**Output:** $\mathbf{x}^*$

1. Set $k = 0$
2. Set $\mathbf{r}_k = \mathbf{A}\mathbf{x}_k - \mathbf{b}$ and $\mathbf{p}_k = -\mathbf{r}_k$
3. **while** $(\mathbf{r}_k \neq 0)$
4. **do** $\alpha_k = -\dfrac{\mathbf{r}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}\mathbf{p}_k}$
5. $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
6. $\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k \mathbf{A}\mathbf{p}_k$
7. $\beta_{k+1} = \dfrac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$
8. $\mathbf{p}_{k+1} = -\mathbf{r}_{k+1} + \beta_{k+1}\mathbf{p}_k$
9. $k = k + 1$
10. end(**while**)
11. **return** $\mathbf{x}^* = \mathbf{x}_k$

The conjugate gradient method exhibits very nice convergence properties. For the sequence $\{\mathbf{x}_k\}$ converging to the solution $\mathbf{x}^*$ of the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ the algorithm takes at most $n$ steps (Theorem 5.1 [NW99]). Furthermore, if the matrix $\mathbf{A}$

has only $r$ distinct eigenvalues, the conjugate gradient iteration will terminate at the solution in at most $r$ iterations (Theorem 5.4 [NW99]). This is true in theory but it has been shown that if the eigenvalues of $\mathbf{A}$ are randomly distributed the rate of convergence is slower. Generally, the conjugate gradient iterates will approximately solve the given problem after $r$ steps if the eigenvalues exist in $r$ distinct clusters (Theorem 5.5 [NW99]). A rough estimate of the convergence rate is given by

$$\frac{\|\mathbf{x}_k - \mathbf{x}^*\|_A}{\|\mathbf{x}_k - \mathbf{x}^*\|_A} \leq \left( \frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1} \right)^{2\kappa}, \tag{3.35}$$

where $\kappa$ is the condition number of the matrix.

## 3.7 Non-Linear Conjugate Gradient Method

The basic idea behind a non-linear conjugate gradient method is similar to that of a conjugate gradient method. Both of these methods generate sets of conjugate search directions. The two methods differ from each other based on nature of the objective function. Conjugate gradient methods are designed to solve systems of linear equations whereas, the non-linear conjugate gradient methods can be used for any general function.

In a non-linear conjugate gradient method, an arbitrary non-linear objective function such as $F$ replaces the convex quadratic function $\phi$ in (3.27). In this setting, exact values of the step size $\alpha_k$ and the coefficient $\beta_{k+1}$ in (3.30) and (3.34), respectively, are no longer valid. Here, the step size $\alpha_k$ is obtained by performing an inexact line search on $F(\mathbf{x}_k + \alpha \mathbf{p}_k)$. The residual $\mathbf{r}_k$ is replaced with the gradient $\nabla F(\mathbf{x}_k)$. This automatically changes the iteration formula for $\beta_{k+1}$. The first formula for computing $\beta_{k+1}$ was proposed by Fletcher and Reeves [FR64] and is given as

$$\beta_{k+1}^{FR} = \frac{\nabla F(\mathbf{x}_{k+1})^T \nabla F(\mathbf{x}_{k+1})}{\nabla F(\mathbf{x}_k)^T \nabla F(\mathbf{x}_k)}. \tag{3.36}$$

Later on, Polak and Ribière [Pol69] suggested a slightly modified version of $\beta_{k+1}^{FR}$, which is

$$\beta_{k+1}^{PR} = \frac{\nabla F(\mathbf{x}_{k+1})^T \big( \nabla F(\mathbf{x}_{k+1}) - \nabla F(\mathbf{x}_k) \big)}{\|\nabla F(\mathbf{x}_k)\|^2}. \tag{3.37}$$

According to (3.32), the formulas for $\beta_{k+1}^{FR}$ and $\beta_{k+1}^{PR}$ are identical if $F$ is quadratic. However, the Polak-Ribière method is more robust and gives faster convergence results in comparison to the Fletcher-Reeves method for general non-linear functions [NW99]. Several other formulas for $\beta_{k+1}$ proposed over the years originate from the work done by Fletcher [Fle87], Liu and Storey [LS91], Dai and Yuan [DY99], Hager and Zhang [HZ05]. An algorithm for the Polak-Ribière non-linear conjugate gradient method works as follows.

**Algorithm** *PR-CG*
**Input:** $\mathbf{x}_0$ and $\epsilon$

**Output:** $\mathbf{x}^*$

1. Set $k = 0$
2. Evaluate $\nabla F(\mathbf{x}_k)$
3. Set $\mathbf{p}_k = -\nabla F(\mathbf{x}_k)$
4. **while** (the Euclidean norm of $\nabla F(\mathbf{x}_k)$ is not sufficiently close to zero specified by the scalar $\epsilon$)
5. **do** Obtain $\alpha_k$ by using the *inexact line search* procedure of Section 3.3.1.2
6. Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
7. Evaluate $\nabla F(\mathbf{x}_{k+1})$
8. $\beta_{k+1}^{PR} = \dfrac{\nabla F(\mathbf{x}_{k+1})^T \big(\nabla F(\mathbf{x}_{k+1}) - \nabla F(\mathbf{x}_k)\big)}{\|\nabla F(\mathbf{x}_k)\|^2}$
9. $\mathbf{p}_{k+1} = -\nabla F(\mathbf{x}_{k+1}) + \beta_{k+1}^{PR}\, \mathbf{p}_k$
10. $k = k + 1$
11. end(**while**)
12. **return** $\mathbf{x}^* = \mathbf{x}_k$

The non-linear conjugate gradient methods may not necessarily converge in $n$ steps. We can improve the convergence rate of the algorithm by introducing certain restart procedures. For example, the iterative non-linear conjugate gradient procedure is restarted by setting $\beta_{k+1}^{PR} = 0$, that is, we take a step in the direction of steepest

descent if the algorithm fails to converge after a fixed number of iterations.

$$\beta_{k+1}^{PR} = \begin{cases} 0, & \text{if } k = cn \text{ for some } c \in \mathbb{N} \\ \dfrac{\nabla F(\mathbf{x}_{k+1})^T \big(\nabla F(\mathbf{x}_{k+1}) - \nabla F(\mathbf{x}_k)\big)}{\|\nabla F(\mathbf{x}_k)\|^2}, & \text{otherwise} \end{cases} \tag{3.38}$$

Another possibility is to restart the iterations when the gradients are far away from being orthogonal, that is,

$$\frac{|\nabla F(\mathbf{x}_k)^T \nabla F(\mathbf{x}_{k-1})|}{\|\nabla F(\mathbf{x}_k)\|^2} \geq \nu. \tag{3.39}$$

A typical value for $\nu$ is 0.1 [NW99]. While solving high-dimensional problems such restarts may never occur. It is often the case that the solution is found in fewer steps than $n$ steps. That is why, the non-linear conjugate gradient methods are mostly recommended for solving high-dimensional problems.

# 4 Previous Work

In this section, we review the methodology adopted by Gutmann and Hyvärinen for optimization of the objective function in noise-contrastive estimation [GH12]. The non-linear conjugate gradient algorithm of Rasmussen [Ras06] is used for optimization which proves to be quite slow in practice for large datasets. To improve the computational performance of the algorithm, random subsets of input samples are used to compute the gradients during the course of optimization. This idea is based on a strategy known as the sample average approximation [KPH15]. As a result, computation time of the algorithm is reduced substantially but the accuracy of the estimates decreases as well. The current optimization strategy leaves room for improvement, as we demonstrate in this section.

## 4.1 Non-Linear Optimization in NCE

Next, we want to focus on the work of Gutmann and Hyvärinen [GH12] for optimization of the objective function,

$$J_T(\boldsymbol{\theta}) = \frac{1}{T_d} \left\{ \sum_{t=1}^{T=T_d+T_n} - C_t \ln h(\mathbf{u}_t; \boldsymbol{\theta}) - (1 - C_t) \ln \big[1 - h(\mathbf{u}_t; \boldsymbol{\theta})\big] \right\}, \tag{4.1}$$

in noise-contrastive estimation discussed in Section 2.2.2. The non-linear conjugate gradient algorithm of Rasmussen [Ras06] was used to minimize $J_T$. The algorithm uses the Polak-Ribière formula in (3.37) to compute $\beta_{k+1}$. The step size $\alpha_k$ is obtained by performing an inexact line search on $J_T(\boldsymbol{\theta}_k + \alpha \mathbf{p}_k)$ which satisfies the Wolfe conditions described in Section 3.3.1.1.

It was shown that by using more noise samples $T_n$ than data samples $T_d$, that is, by increasing $\nu$, more accurate estimates can be obtained [GH12]. However, using a large value for $\nu$ slows down the optimization process because more and more noise samples need to be processed. Thus, there exists a trade-off between statistical accuracy of the estimates and computational performance of the algorithm.

## 4.2 Sample Average Approximation (SAA)

In order to obtain better convergence results, Rasmussen's algorithm was modified to stochastically approximate the gradient of $J_T$ instead of using the exact gradient. The idea is to select random subsets of input samples (data and noise) to approximate the gradient vectors instead of using whole sets of input samples. The objective function $J_T$ can be written as an expectation over the input samples as follows

$$J_T(\boldsymbol{\theta}) = \frac{1}{T_d} \sum_{t=1}^{T=T_d+T_n} J_t(\mathbf{u}_t; \boldsymbol{\theta}), \tag{4.2}$$

and the exact gradient of $J_T$ at the iterate $\boldsymbol{\theta}_k$ is computed as

$$\nabla J_T(\boldsymbol{\theta}_k) = \frac{1}{T_d} \sum_{t=1}^{T=T_d+T_n} \nabla J_t(\mathbf{u}_t; \boldsymbol{\theta}_k). \tag{4.3}$$

In every *k-th* iteration, a random subset of input samples, say $S_k \subset \mathcal{U}$, is selected such that $|S_k| < |\mathcal{U}|$. We use $\nabla \widetilde{J}(\boldsymbol{\theta}_k)$ to denote the approximation to the gradient given as

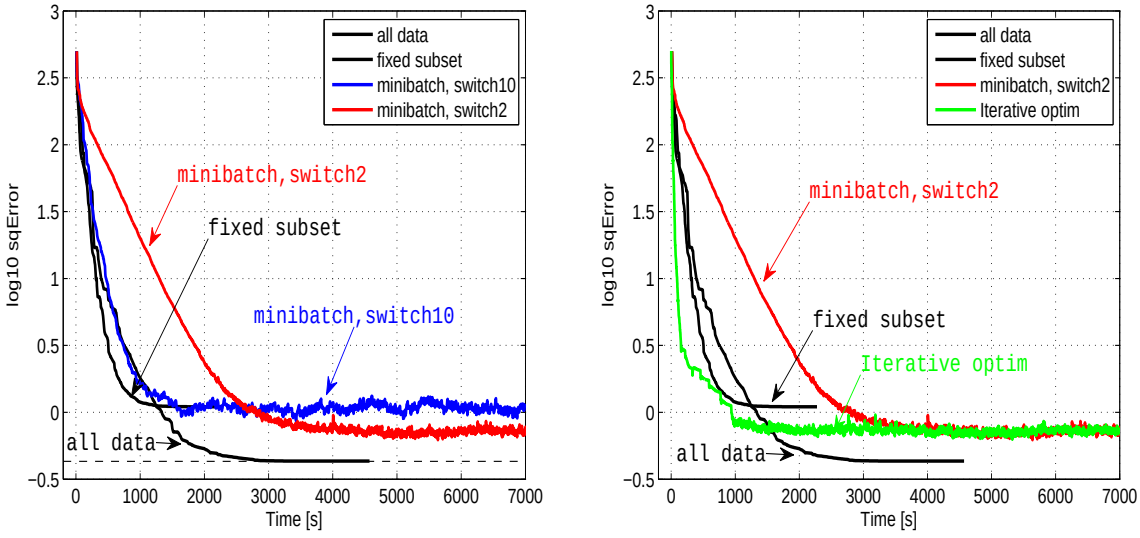
$$\nabla \widetilde{J}_{S_k}(\boldsymbol{\theta}_k) = \frac{1}{T_d} \sum_{\mathbf{u} \in S_k} \nabla J(\mathbf{u}; \boldsymbol{\theta}_k). \tag{4.4}$$

As fewer samples are being used to compute the gradient therefore, each iteration should take less time. Ideally, the function in (4.4) is expected to behave approximately like (4.3) despite the errors induced by the noisy gradient. As a result, the

convergence rate of the algorithm should improve while accuracy of the estimates remains intact.

In Figure 6a, the black curve represents a situation where all data is used leading to the smallest error which can be obtained with noise-contrastive estimation for $\nu = 10$ and $T_d = 50,000$. The upper black curve in the same figure shows the mean squared error (MSE) if only a fixed subset with $\widetilde{T}_d = 25,000$ is used for optimization. The red curve shows an improved performance, if a random subset with $\widetilde{T}_d$ is selected after every two updates of the parameters. In this case, the algorithm converges faster but results in less accurate estimates. Selecting a random subset with $\widetilde{T}_d$ after every ten updates of the parameters yields only minor improvements.

Figure 6b shows the results of a different optimization strategy. The objective function $J_T$ is iteratively optimized for increasingly large values of $\nu$. Whenever, $\nu$ is increased to $\nu + 1$, a new subset is selected. This subset is switched after every 50 iterations until $\nu$ reaches its maximal value. Here, a maximal value of $\nu = 10$ is used and the subset is switched after two parameter updates. We can see that this strategy results in faster convergence but still leads to the same estimates as indicated by the red curve.




(a) Increasing accuracy

(b) Increasing accuracy and speed of convergence

Figure 6: Analysis of optimization strategy (Figure 20 in Appendix C of [GH12], reproduced with permission of the authors).

## 4.3 Limitations of the Current Strategy

We now summarize the key observations related to the non-linear optimization in noise-contrastive estimation as follows

- Larger values for the parameter $\nu$ give better estimation results. However, using more and more noise samples slows down the optimization process.
- The computational performance of the algorithm improves if fewer samples are used to compute the gradients.
- The algorithm converges much faster if a fixed subset is used. However, a single subset does not adequately represent the overall picture of the function $J_T$ and results in less precise estimates (upper black curves in Figures 6a and 6b).
- When the subsets are switched repeatedly, all data is actually used in the optimization leading to better estimates (red curves in Figures 6a and 6b). However, there is an increased overhead because with every new subset the algorithm redirects from its current path of optimization.
- Even with the best strategy so far, the mean squared error is still higher than the baseline (green curve in Figure 6b). This is due to the fact that the non-linear Polak-Ribière conjugate gradient method does not overcome the errors induced by the stochastically approximated gradient. We explain this phenomenon below:

In presence of a noisy gradient, convergence is guaranteed if diminishing step sizes such as $\alpha_k = \mathcal{O}\left(\frac{1}{k}\right)$ are used [BT99]. In this situation, noise in the gradient is smoothed out over the iterations providing an accurate estimate at the time of convergence. It means that the price of achieving an accurate solution is a very slow convergence rate. Now consider the very first iteration of the Polak-Ribière method implemented using stochastic approximation to the gradient of $J_T$, the initial search direction is $\mathbf{p}_0 = -\nabla \widetilde{J}_{S_0}(\boldsymbol{\theta}_0)$. The step size $\alpha_0$ is obtained by solving the following minimization problem inexactly,

$$\min_{\alpha \in \mathbb{R}} J_T\big(\boldsymbol{\theta}_0 - \alpha \nabla \widetilde{J}_{S_0}(\boldsymbol{\theta}_0)\big); \qquad \alpha > 0. \tag{4.5}$$

The resulting value $\alpha_0$ minimizes $J_T$ in the direction of $-\nabla \widetilde{J}_{S_0}(\boldsymbol{\theta}_0)$ rather than in the steepest descent direction indicated by $-\nabla J_T(\boldsymbol{\theta}_0)$. As the steps taken by the Rasmussen's algorithm are much larger, the error induced by the noisy gradient is perpetrated through all the iterations and thus leads to less accurate estimates.

# 5 Research Goals

The main goal of this thesis is to implement computationally efficient optimization methods for optimization of the objective function in noise-contrastive estimation that do not compromise the accuracy of the estimates.

First-order optimization methods like, for example, the non-linear conjugate gradient method [Pol69], require minimal information to make progress. Value and gradient of the objective function are computed iteratively until the algorithm converges. These methods are easy to implement. However, they exhibit a linear rate of convergence and tend to be slow in producing high accuracy results. Furthermore, these methods generate poor search directions in presence of a noisy gradient leading to less accurate estimates. Unless, we use diminishing step sizes which have an adverse effect on the computational performance of an algorithm.

In order to achieve our research goals, we propose to use second-order optimization methods based on the Newton method [DS96]. This method uses both the gradient and the Hessian matrix to produce better search directions and converges much more quickly. Another reason to use second-order methods is the fact that they have been shown to produce good results when incorporated with stochastic approximation to the Hessian [BCNN11]. The Newton based optimization methods will produce reliable search directions as long as the Hessian or the approximation matrix used instead of the Hessian is positive definite. The Newton method, when close to a minimum, converges quite rapidly. But if it is started far away from the minimum, it can take a long time to converge and, in fact move to a totally different region of the parameter space. The Newton method needs to be modified to produce solutions converging to a minimum. In this thesis, we use two of the Newton-based optimization methods known as the line-search Newton-CG and the trust region Newton-CG methods [NW99].

# 6 Newton-based Optimization Methods for NCE

We begin this section by highlighting some of the key issues associated with the Newton method discussed in Section 3.5. This is followed by explanations of the line search Newton-CG and the trust region Newton-CG methods.

## 6.1 Limitations of the Newton Method

The Newton method needs to be modified for non-linear optimization in noise-contrastive estimation because of the following two main reasons.

**1. The matrix $\mathbf{H}_k$ (Hessian or an approximation matrix) must be non-singular and positive definite at all times.**

The Newton step has the form

$$\mathbf{p}_k = -\big[\mathbf{H}_k\big]^{-1} \nabla F(\mathbf{x}_k). \tag{6.1}$$

Given a non-zero gradient vector $\nabla F(\mathbf{x}_k) \neq 0$, the Newton step $\mathbf{p}_k$ must be a descent direction, that is,

$$\nabla F(\mathbf{x}_k)^T \, \mathbf{p}_k = -\nabla F(\mathbf{x}_k)^T \left[\mathbf{H}_k\right]^{-1} \nabla F(\mathbf{x}_k) < 0 \tag{6.2}$$

or

$$\nabla F(\mathbf{x}_k)^T \left[\mathbf{H}_k\right]^{-1} \nabla F(\mathbf{x}_k) > 0, \tag{6.3}$$

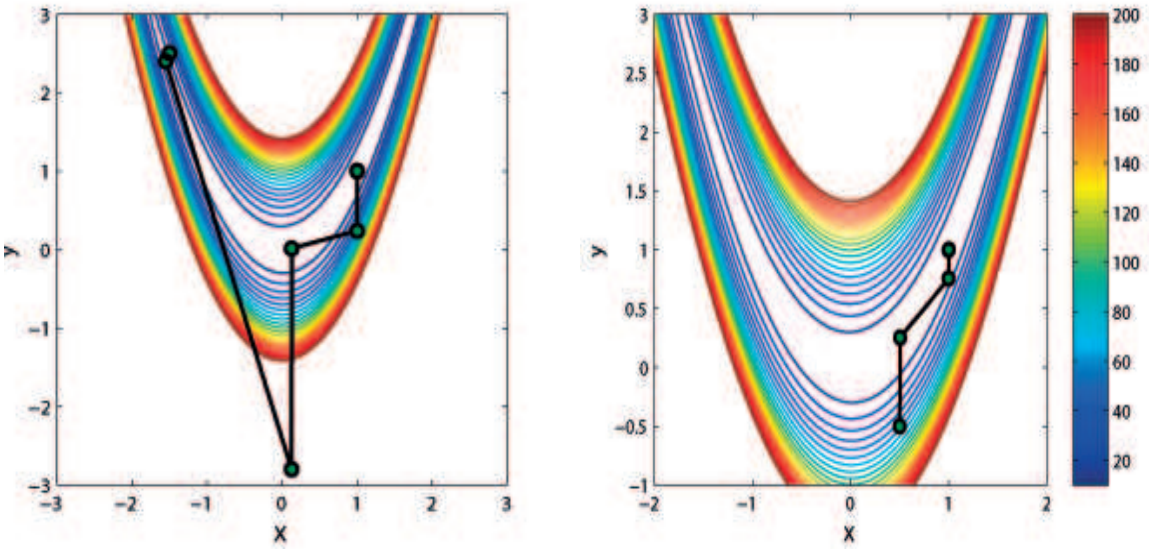
must be satisfied. The above inequality holds if and only if the matrix $\mathbf{H}_k$ is non-singular and positive definite at all times.

**Solution:** In noise-contrastive estimation, sometimes the Hessian $\nabla^2 J_T(\boldsymbol{\theta}_k)$ may not be positive definite which would cause the Newton step to be an ascent direction. Therefore, we use *Gauss-Newton approximation* [Che11] to the Hessian. This approximation matrix is always at least positive semi-definite. We describe a procedure to compute the Gauss-Newton approximation to the Hessian of $J_T$ in Section 6.2.

**2. There is no guarantee that the Newton method reduces the function value with each step.**

The Newton method takes unit steps to move from the current iterate $\mathbf{x}_k$ to the next iterate $\mathbf{x}_{k+1}$. In case of the non-linear optimization, unit steps do not guarantee a regular decrease in the function values, that is, $F(\mathbf{x}_k + \alpha_k \mathbf{p}_k) < F(\mathbf{x}_k)$. We show this in Figure 7. In the absence of any proper adjustment, the algorithm may require a large number of iterations to converge. In some cases, the algorithm may never attain convergence especially if the starting point lies far away from the solution. Even if $\mathbf{H}_k$ is always non-singular, the algorithm may not converge, unless started close enough to the solution.

**Solution:** We use the line search and the trust region strategies to control for the size of the Newton step. As a result, a sequence of non-zero step sizes $\{\alpha_k\}$ is generated which guarantees that there is a systematic decrease in the function values, that is, $F(\mathbf{x}_k + \alpha_k \mathbf{p}_k) < F(\mathbf{x}_k)$, throughout the course of optimization. The Newton based optimization methods known as the line search Newton-CG and the trust region Newton-CG methods are described in Sections 6.4 and 6.5 respectively.




(a) (b)

Figure 7: Minimization of the Rosenbrock function $f(x, y) = 100(y - x^2)^2 + (1 - x)^2$ using the Newton method of Section 3.5. The function $f(x, y)$ attains its minimum value at $(\mathbf{x}^*, \mathbf{y}^*) = (1, 1)$. Different initial guesses $(\mathbf{x}_0^1, \mathbf{y}_0^1) = (-1.5, 2.5)$ and $(\mathbf{x}_0^2, \mathbf{y}_0^2) = (0.5, -0.5)$ are used in (a) and (b) respectively. Unit step sizes, that is, $\alpha_k = 1$ do not produce a regular decrease in the function values.

## 6.2 Gauss-Newton Approximation

First of all, we need to derive closed-form expressions for the gradient and the Hessian of the objective function,

$$J_T(\boldsymbol{\theta}) = \frac{1}{T_d} \sum_{t=1}^{T_d+T_n} J(\mathbf{u}_t; \boldsymbol{\theta}); \qquad \mathbf{u}_t \in \mathbb{R}^n \text{ and } \boldsymbol{\theta} \in \mathbb{R}^d, \tag{6.4}$$

in noise-contrastive estimation (see Section 2.2.2). The function $J(\mathbf{u}_t;\boldsymbol{\theta})$ is of the form

$$J\big(\mathbf{u}_t;\boldsymbol{\theta}\big) = -\,C_t\,\ln h\big(\mathbf{u}_t;\boldsymbol{\theta}\big) - \big(1-C_t\big)\ln\Big(1-h\big(\mathbf{u}_t;\boldsymbol{\theta}\big)\Big), \tag{6.5}$$

where

$$h\big(\mathbf{u}_t;\boldsymbol{\theta}\big) = \frac{1}{1+\nu\exp\Big(-G\big(\mathbf{u}_t;\boldsymbol{\theta}\big)\Big)}, \tag{6.6}$$

and

$$G\big(\mathbf{u}_t;\boldsymbol{\theta}\big) = \ln p_m\big(\mathbf{u}_t;\boldsymbol{\theta}\big) - \ln f_n\big(\mathbf{u}_t\big). \tag{6.7}$$

We use the following shorthand notations for future references,

$$J_t\big(\boldsymbol{\theta}\big) = J(\mathbf{u}_t;\boldsymbol{\theta}); \qquad h_t\big(\boldsymbol{\theta}\big) = h\big(\mathbf{u}_t;\boldsymbol{\theta}\big); \qquad G_t\big(\boldsymbol{\theta}\big) = G\big(\mathbf{u}_t;\boldsymbol{\theta}\big). \tag{6.8}$$

The gradient of $J_T$ is defined as an average over multiple gradient vectors, that is,

$$\nabla J_T\big(\boldsymbol{\theta}\big) = \frac{1}{T_d}\sum_{t=1}^{T_d+T_n}\nabla J_t\big(\boldsymbol{\theta}\big), \tag{6.9}$$

The gradient vector computed at $\mathbf{u}_t$ has the form

$$\nabla J_t\big(\boldsymbol{\theta}\big) = \frac{\partial J_t\big(\boldsymbol{\theta}\big)}{\partial\boldsymbol{\theta}} = \begin{pmatrix}\dfrac{\partial J_t\big(\boldsymbol{\theta}\big)}{\partial\theta_1} & \dfrac{\partial J_t\big(\boldsymbol{\theta}\big)}{\partial\theta_2} & \cdots\cdots & \dfrac{\partial J_t\big(\boldsymbol{\theta}\big)}{\partial\theta_d}\end{pmatrix}^T. \tag{6.10}$$

Since $J_t$ is a composite function, its partial derivatives are obtained as follows:

$$\frac{\partial J_t\big(\boldsymbol{\theta}\big)}{\partial\theta_i} = \frac{\partial J_t\big(\boldsymbol{\theta}\big)}{\partial h_t\big(\boldsymbol{\theta}\big)}\frac{\partial h_t\big(\boldsymbol{\theta}\big)}{\partial G_t\big(\boldsymbol{\theta}\big)}\frac{\partial G_t\big(\boldsymbol{\theta}\big)}{\partial\theta_i}; \qquad i=1,2,\ldots,d. \tag{6.11}$$

$$\begin{aligned}(i)\quad \frac{\partial J_t\big(\boldsymbol{\theta}\big)}{\partial h_t\big(\boldsymbol{\theta}\big)} &= -\frac{C_t}{h_t\big(\boldsymbol{\theta}\big)}\cdot(1) - \frac{1-C_t}{1-h_t\big(\boldsymbol{\theta}\big)}\cdot(-1),\\ &= -\frac{C_t}{h_t\big(\boldsymbol{\theta}\big)} + \frac{1-C_t}{1-h_t\big(\boldsymbol{\theta}\big)},\\ &= \frac{-C_t + C_t h_t\big(\boldsymbol{\theta}\big) + h_t\big(\boldsymbol{\theta}\big) - C_t h_t\big(\boldsymbol{\theta}\big)}{h_t\big(\boldsymbol{\theta}\big)\Big(1-h_t\big(\boldsymbol{\theta}\big)\Big)},\\ &= \frac{\Big(h_t\big(\boldsymbol{\theta}\big)-C_t\Big)}{h_t\big(\boldsymbol{\theta}\big)\Big(1-h_t\big(\boldsymbol{\theta}\big)\Big)}.\end{aligned} \tag{6.12}$$

$$
\begin{aligned}
(ii)\quad \frac{\partial h_t(\boldsymbol{\theta})}{\partial G_t(\boldsymbol{\theta})} &= -\Big[1+\nu\exp\Big(-G\big(\mathbf{u}_t;\boldsymbol{\theta}\big)\Big)\Big]^{-2}\cdot\Big[\nu\exp\Big(-G\big(\mathbf{u}_t;\boldsymbol{\theta}\big)\Big)\Big]\cdot(-1),\\
&= \frac{\Big[\nu\exp\Big(-G\big(\mathbf{u}_t;\boldsymbol{\theta}\big)\Big)\Big]}{\Big[1+\nu\exp\Big(-G\big(\mathbf{u}_t;\boldsymbol{\theta}\big)\Big)\Big]^2},\\
&= \frac{1}{1+\nu\exp\Big(-G\big(\mathbf{u}_t;\boldsymbol{\theta}\big)\Big)}\cdot\frac{\nu\exp\Big(-G\big(\mathbf{u}_t;\boldsymbol{\theta}\big)\Big)}{1+\nu\exp\Big(-G\big(\mathbf{u}_t;\boldsymbol{\theta}\big)\Big)},\\
&= h_t(\boldsymbol{\theta})\Big(1-h_t(\boldsymbol{\theta})\Big). \qquad (6.13)
\end{aligned}
$$

$(iii)$ The term $\dfrac{\partial G_t(\boldsymbol{\theta})}{\partial \theta_i}$ depends on the data and the noise distributions of a given model. We derive it separately for the unnormalized Gaussian and the unnormalized independent component analysis (ICA) models in Sections 7.1 and 7.2 respectively.

Using (6.12) and (6.13), we can update (6.11) as follows:

$$
\begin{aligned}
\frac{\partial J_t(\boldsymbol{\theta})}{\partial \theta_i} &= \frac{\Big(h_t(\boldsymbol{\theta})-C_t\Big)}{h_t(\boldsymbol{\theta})\Big(1-h_t(\boldsymbol{\theta})\Big)}\cdot h_t(\boldsymbol{\theta})\Big(1-h_t(\boldsymbol{\theta})\Big)\cdot\frac{\partial G_t(\boldsymbol{\theta})}{\partial \theta_i},\\
&= \Big(h_t(\boldsymbol{\theta})-C_t\Big)\cdot\frac{\partial G_t(\boldsymbol{\theta})}{\partial \theta_i}, \qquad (6.14)
\end{aligned}
$$

that is, in vector notation,

$$\nabla J_t(\boldsymbol{\theta}) = \Big(h_t(\boldsymbol{\theta})-C_t\Big)\,\nabla_{\boldsymbol{\theta}} G_t. \qquad (6.15)$$

Finally, the gradient of $J_T$ is

$$\boxed{\nabla J_T(\boldsymbol{\theta}) = \frac{1}{T_d}\sum_{t}^{T_d+T_n}\Big(h_t(\boldsymbol{\theta})-C_t\Big)\,\nabla_{\boldsymbol{\theta}} G_t.} \qquad (6.16)$$

Similarly, the Hessian of $J_T$ is also obtained as an expectation, that is,

$$\nabla^2 J_T(\boldsymbol{\theta}) = \frac{1}{T_d}\sum_{t=1}^{T_d+T_n}\nabla^2 J_t(\boldsymbol{\theta}). \qquad (6.17)$$

Each entry of the matrix $\nabla^2 J_t(\boldsymbol{\theta})$ is of the form,

$$\big[\nabla^2 J_t(\boldsymbol{\theta})\big]_{ij} = \frac{\partial^2 J_t(\boldsymbol{\theta})}{\partial\theta_i\partial\theta_j}; \qquad i,j=1,2,\ldots,d. \qquad (6.18)$$

Using (6.14), we have

$$\begin{aligned}\big[\nabla^2 J_t(\boldsymbol{\theta})\big]_{ij} &= \frac{\partial}{\partial\theta_i}\bigg(\Big(h_t(\boldsymbol{\theta}) - C_t\Big)\cdot\frac{\partial G_t(\boldsymbol{\theta})}{\partial\theta_j}\bigg),\\ &= \frac{\partial\Big(h_t(\boldsymbol{\theta}) - C_t\Big)}{\partial\theta_i}\cdot\frac{\partial G_t(\boldsymbol{\theta})}{\partial\theta_j} + \Big(h_t(\boldsymbol{\theta}) - C_t\Big)\cdot\frac{\partial^2 G_t(\boldsymbol{\theta})}{\partial\theta_i\partial\theta_j}. \quad (6.19)\end{aligned}$$

We discussed in Section 6.1 that the Newton method requires the Hessian to be positive definite in order to make progress. The Hessian in noise-contrastive estimation may not always satisfy this criteria. In this thesis, we use Gauss-Newton approximation to the Hessian of $J_T$, say $B_T(\boldsymbol{\theta})$, which is always at least positive semi-definite. It is obtained by dropping the second-order derivatives [Che11]. We continue our calculations from (6.19),

$$\begin{aligned}\big[B_t(\boldsymbol{\theta})\big]_{ij} &= \frac{\partial\Big(h_t(\boldsymbol{\theta}) - C_t\Big)}{\partial\theta_i}\cdot\frac{\partial G_t(\boldsymbol{\theta})}{\partial\theta_j} + 0,\\ &= \left(\frac{\partial h_t(\boldsymbol{\theta})}{\partial\theta_i} - \frac{\partial C_t}{\partial\theta_i}\right)\cdot\frac{\partial G_t(\boldsymbol{\theta})}{\partial\theta_j},\\ &= \left(\frac{\partial h_t(\boldsymbol{\theta})}{\partial\theta_i} + 0\right)\cdot\frac{\partial G_t(\boldsymbol{\theta})}{\partial\theta_j},\\ &= \frac{\partial h_t(\boldsymbol{\theta})}{\partial G_t(\boldsymbol{\theta})}\cdot\frac{\partial G_t(\boldsymbol{\theta})}{\partial\theta_i}\cdot\frac{\partial G_t(\boldsymbol{\theta})}{\partial\theta_j}.\end{aligned}$$

Using the value of $\dfrac{\partial h_t(\boldsymbol{\theta})}{\partial G_t(\boldsymbol{\theta})}$ from (6.13), we have

$$\big[B_t(\boldsymbol{\theta})\big]_{ij} = h_t(\boldsymbol{\theta})\Big(1 - h_t(\boldsymbol{\theta})\Big)\cdot\frac{\partial G_t(\boldsymbol{\theta})}{\partial\theta_i}\cdot\frac{\partial G_t(\boldsymbol{\theta})}{\partial\theta_j}. \quad (6.20)$$

Hence, the Gauss-Newton approximation to the Hessian of $J_T$ is given as

$$\boxed{\nabla^2 J_T(\boldsymbol{\theta}) \approx \mathbf{B}_T(\boldsymbol{\theta}) = \frac{1}{T_d}\sum_{t=1}^{T_d+T_n} h_t(\boldsymbol{\theta})\Big(1 - h_t(\boldsymbol{\theta})\Big)\nabla_{\boldsymbol{\theta}} G_t \nabla_{\boldsymbol{\theta}} G_t^T.} \quad (6.21)$$

If the vector of partial derivatives $\nabla_{\boldsymbol{\theta}} G_t$ is independent of the parameter vector $\boldsymbol{\theta}$, then the exact Hessian of $J_T$ is equal to its Gauss-Newton approximation, that is, $\nabla^2 J_T\big(\boldsymbol{\theta}\big) = B_T(\boldsymbol{\theta})$. We show this in Section 7.1. To show that the Gauss-Newton

approximation is always at least positive semi-definite, consider

$$\begin{aligned}\mathbf{v}^T B_T(\boldsymbol{\theta})\mathbf{v} &= \frac{1}{T_d}\sum_{t=1}^{T_d+T_n} h_t\big(\boldsymbol{\theta}\big)\Big(1-h_t\big(\boldsymbol{\theta}\big)\Big)\mathbf{v}^T\nabla_{\boldsymbol{\theta}}G_t\nabla_{\boldsymbol{\theta}}G_t^T\mathbf{v}; \quad \text{for} \quad \mathbf{v}\neq 0,\\ &= \frac{1}{T_d}\sum_{t=1}^{T_d+T_n} h_t\big(\boldsymbol{\theta}\big)\Big(1-h_t\big(\boldsymbol{\theta}\big)\Big)\left\|\mathbf{v}^T\nabla_{\boldsymbol{\theta}}G_t\right\|^2,\\ &\geq 0. \end{aligned} \tag{6.22}$$

The function $h_t\big(\boldsymbol{\theta}\big)$ denotes the posterior probability of the target class $C_t = 1$ given the input sample $\mathbf{u}_t$ and the parameter vector $\boldsymbol{\theta}$, that is, $0 \leq h_t\big(\boldsymbol{\theta}\big) \leq 1$. The product $\left\|\mathbf{v}^T\nabla_{\boldsymbol{\theta}}G_t\right\|^2$ is generally a non-negative quantity. The matrix $B_T(\boldsymbol{\theta})$ is always positive definite unless $h_t\big(\boldsymbol{\theta}\big) = 0$ or $h_t\big(\boldsymbol{\theta}\big) = 1$ for all $t$ in which case it is positive semi-definite. Other possible scenario when $B_T(\boldsymbol{\theta})$ could be positive semi-definite is if $\nabla_{\boldsymbol{\theta}}G_t$ but this is never the case. Hence, we now have a working Hessian. Unless stated otherwise, we always use the Gauss-Newton approximation to the Hessian of $J_T$ for the estimation of unnormalized models in this thesis.

## 6.3 Inexact Newton Step

The first thing that we do is model the objective function in noise-contrastive estimation using second-order Taylor series expansion centered at the current iterate $\boldsymbol{\theta}_k$ as follows

$$J_T(\boldsymbol{\theta}_k + \mathbf{p}) \approx m_k(\mathbf{p}) = J_T(\boldsymbol{\theta}_k) + \nabla J_T(\boldsymbol{\theta}_k)^T\mathbf{p} + \frac{1}{2}\,\mathbf{p}^T B_T(\boldsymbol{\theta}_k)\mathbf{p}, \tag{6.23}$$

where $\nabla J_T(\boldsymbol{\theta}_k)$ is the gradient and $B_T(\boldsymbol{\theta}_k)$ is the Gauss-Newton approximation to the Hessian of $J_T$. The exact Newton step $\mathbf{p}_k$ is obtained by setting the gradient of the quadratic model $m_k$ to zero. By doing so, we assume that $m_k$ provides an accurate representation of $J_T$ throughout the course of optimization. This quadratic approximation may not be viable if the current iterate is located far away from the solution. In such situations, the cost of obtaining an exact Newton step will be just too high. In practice, search for the Newton step is terminated when the gradient of the quadratic model has been reduced to a significant level. This heuristic is implemented as an iterative procedure. The iterative solver is terminated when

$$\|\,\mathbf{r}_k\| \leq \eta_k\|\,\nabla J_T(\boldsymbol{\theta}_k)\|; \qquad 0 < \eta_k < 1, \tag{6.24}$$

where $\mathbf{r}_k$ is the residual of $\nabla J_T(\boldsymbol{\theta}_k + \mathbf{p}) = 0$ and is given as

$$\mathbf{r}_k = B_T(\boldsymbol{\theta}_k)\,\widetilde{\mathbf{p}}_k + \nabla J_T(\boldsymbol{\theta}_k). \tag{6.25}$$

Here, $\widetilde{\mathbf{p}}_k$ is the inexact Newton step. The sequence $\{\eta_k\}$ is known as the *forcing sequence*. It is a key in studying the convergence properties of the inexact Newton methods. It has been shown that by setting $\eta_k = \min(0.5, \sqrt{\| \nabla J_T(\boldsymbol{\theta}_k)\|})$ super-linear convergence is obtained and that $\eta_k = \min(0.5, \| \nabla J_T(\boldsymbol{\theta}_k)\|)$ yields quadratic convergence (see Theorem 6.1 and Theorem 6.2 [NW99]). These results are local in nature. It is assumed that the sequence $\{\boldsymbol{\theta}_k\}$ eventually enters into the neighborhood of $\boldsymbol{\theta}^*$ and that unit step sizes are always used.

## 6.4 Line Search Newton-CG Method

The first Newton-based optimization method used in this thesis is the line search Newton-CG method also known as the truncated Newton method [NW99]. This method computes the inexact Newton step $\widetilde{\mathbf{p}}_k$ using the conjugate gradient method of Section 3.6 and the step size $\alpha_k$ is obtained by performing inexact line search along the line $J_T(\boldsymbol{\theta}_k + \alpha\widetilde{\mathbf{p}}_k)$ as discussed in Section 3.3.1.2. Both the procedures require some minor changes in their implementations to suit our current needs. The Newton equation derived from the quadratic model $m_k$ in (6.23) is given as

$$B_T(\boldsymbol{\theta}_k)\,\widetilde{\mathbf{p}}_k = -\nabla J_T(\boldsymbol{\theta}_k), \tag{6.26}$$

which forms a system of linear equations. We use the conjugate gradient method to compute $\widetilde{\mathbf{p}}_k$ with the following modifications.

- To keep the current implementation of the conjugate gradient algorithm intact, we always use the relations $\mathbf{A} = B_T(\boldsymbol{\theta}_k), \mathbf{x} = \widetilde{\mathbf{p}}_k$ and $\mathbf{b} = -\nabla J_T(\boldsymbol{\theta}_k)$.
- We set $\epsilon = \eta_k$ where $\eta_k = \min(0.5, \| \nabla J_T(\boldsymbol{\theta}_k)\|)$ or $\eta_k = \min(0.5, \sqrt{\| \nabla J_T(\boldsymbol{\theta}_k)\|})$.
- A zero vector is used as an initial guess, that is, $\mathbf{x}^{(0)} = 0$.
- We return the final iterate $\mathbf{x}^{(*)}$ as the inexact Newton step $\widetilde{\mathbf{p}}_k$.

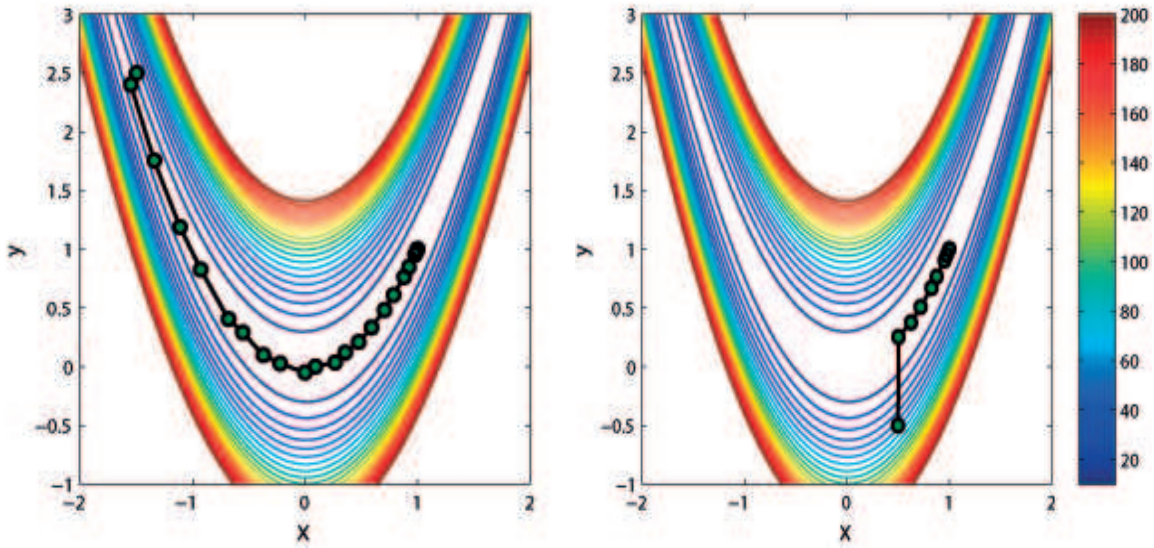
Please note, that the sequence $\{\mathbf{x}^{(i)}\}$ denote the iterates generated by the conjugate gradient algorithm. Secondly, we always use a unit step size in the initial phase of the inexact line search procedure. This preserves the quadratic convergence rate of the Newton method once we enter into the neighborhood of $\boldsymbol{\theta}^*$. An algorithm for the line search Newton-CG method is as follows:

**Algorithm** *LSNewton-CG*
**Input:** $\boldsymbol{\theta}_0$

**Output:** $\boldsymbol{\theta}^*$

1. **for** $(k = 0, 1, 2, \dots)$
2. **do** Evaluate $\nabla J_T(\boldsymbol{\theta}_k)$ and $B_T(\boldsymbol{\theta}_k)$
3. Set $\eta_k = \min(0.5, \|\nabla J_T(\boldsymbol{\theta}_k)\|)$ or $\eta_k = \min(0.5, \sqrt{\|\nabla J_T(\boldsymbol{\theta}_k)\|})$
4. $\widetilde{\mathbf{p}}_k = \textbf{ConjugateGradient}(0, B_T(\boldsymbol{\theta}_k), -\nabla J_T(\boldsymbol{\theta}_k), \eta_k)$
5. Obtain $\alpha_k$ by using the **inexact line search** procedure of Section 3.3.1.2
6. $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha_k \widetilde{\mathbf{p}}_k$
7. end(**for**)
8. **return** $\boldsymbol{\theta}^* = \boldsymbol{\theta}_k$

Figure 8 illustrates the minimization of the Rosenbrock function. Size of the inexact Newton step is controlled using the inexact line search procedure. We can see that now there is a regular decrease in the function values irrespective of choice of the starting point.




(a) (b)

Figure 8: Minimization of the Rosenbrock function $f(x, y) = 100(y - x^2)^2 + (1 - x)^2$ using the line search Newton-CG method.

## 6.5 Trust Region Newton-CG Method

The trust region Newton-CG method [NW99] forms a region around the current iterate which restricts the norm of the Newton step to be no greater than the radius of the trust region. We presented the algorithm for adjusting the size of the trust region in Section 3.3.2. Now, we provide a solution to the constrained sub-problem:

$$\min_{\mathbf{p}\in\mathbb{R}^n} J_T(\boldsymbol{\theta}_k) + \nabla J_T(\boldsymbol{\theta}_k)^T\mathbf{p} + \frac{1}{2}\,\mathbf{p}^T B_T(\boldsymbol{\theta}_k)\mathbf{p} \quad \text{s.t.} \parallel \mathbf{p}\parallel \leq \Delta_k, \tag{6.27}$$

where $\Delta_k$ is the radius of the trust region. Here, the search direction $\mathbf{p}_k$ is not identical to the inexact Newton step defined in Section 6.3. In fact, it is inexact in a way that its norm is restricted by the value of $\Delta_k$. In order to solve for $\mathbf{p}_k$ we cannot directly use the conjugate gradient method since, it is designed to return the unconstrained solution of the system of linear equations. Instead, we use a solution proposed by Steihaug [Ste86] having its roots in the conjugate gradient method. To make a connection between the Steihaug's algorithm and the conjugate gradient algorithm of Section 3.6, please note that

- The quadratic model in (6.27) takes the place of the function $\phi(.)$ in 3.27.
- We set $\mathbf{A} = B_T(\boldsymbol{\theta}_k)$ and $\mathbf{b} = -J_T(\boldsymbol{\theta}_k)$ and initialize $\mathbf{p}_0$ to 0.
- The final iterate $\mathbf{p}^*$ is the Newton step $\mathbf{p}_k$.

Steihaug's approach can be stated formally as follows:

**Algorithm** *CG-Steihaug*
**Input:** $J_T(\boldsymbol{\theta}_k),\ B_T(\boldsymbol{\theta}_k),\ \Delta_k$ and $\epsilon$

**Output:** $\mathbf{p}^*$

1. Set $\Delta = \Delta_k,\ \mathbf{p}_0 = 0,\ \mathbf{r}_0 = \nabla J_T(\boldsymbol{\theta}_k),\ \mathbf{B} = \mathbf{B}_T(\boldsymbol{\theta}_k)$ and $\mathbf{d} = -\mathbf{r}_0$
2. **for** $j = 0, 1, 2, \ldots$
3. Set $\alpha_j = \dfrac{\mathbf{r}_j^T\mathbf{r}_j}{\mathbf{d}_j^T\mathbf{B}\mathbf{d}_j}$
4. Set $\mathbf{p}_{j+1} = \mathbf{p}_j + \alpha_j\mathbf{d}_j$
5. **if** $\|\mathbf{p}_{j+1}\| \geq \Delta$
6. Find $\tau$ such that $\mathbf{p} = \mathbf{p}_j + \tau\mathbf{d}_j$ satisfies $\|\mathbf{p}\| = \Delta$
7. Set $\mathbf{p}^* = \mathbf{p}$ and **stop**

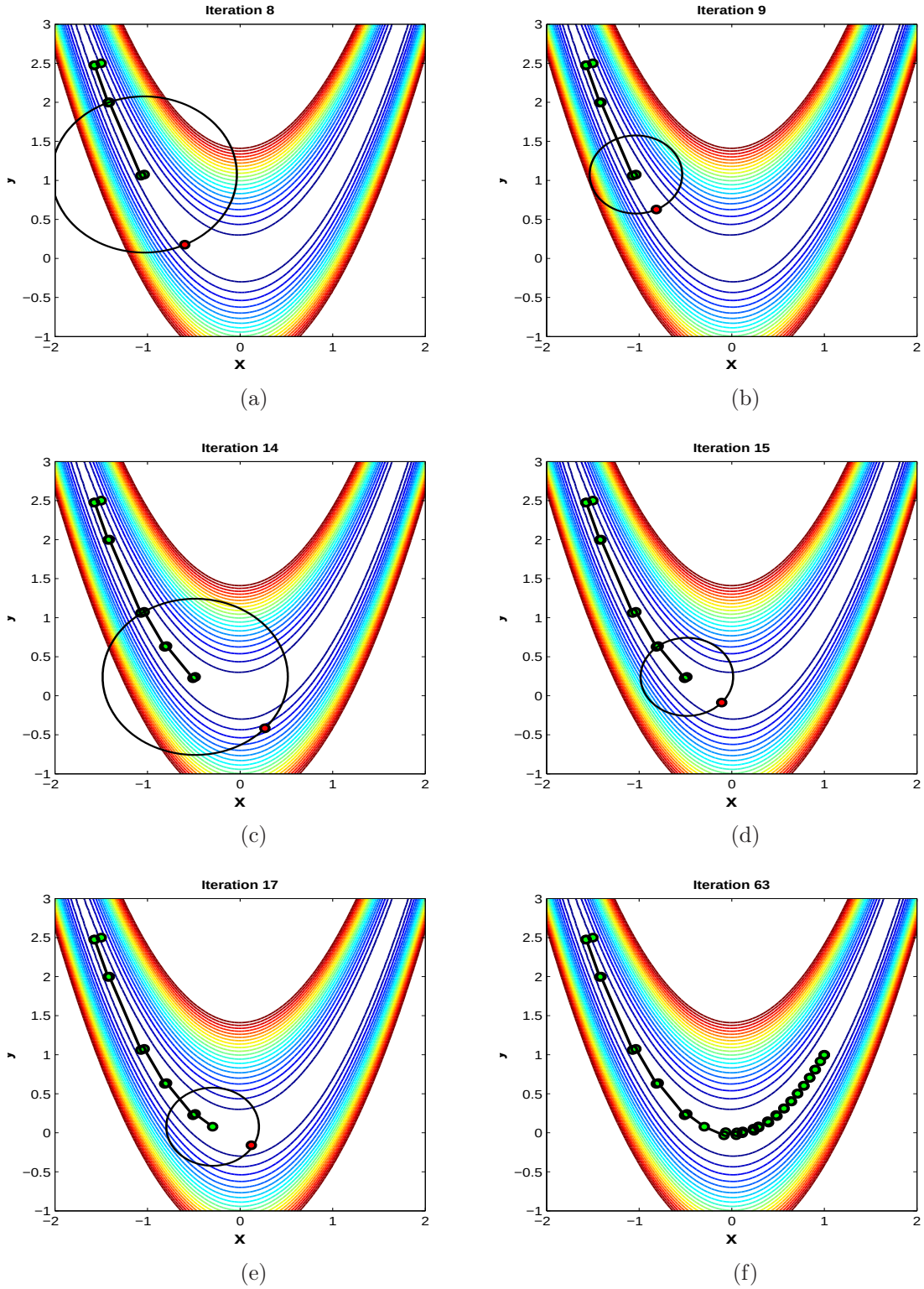
8. end(**if**)
9. Set $\mathbf{r}_{j+1} = \mathbf{r}_j + \alpha_j \mathbf{B}\mathbf{d}_j$
10. **if** $\|\mathbf{r}_{j+1}\| < \epsilon \|\mathbf{r}_0\|$
11. Set $\mathbf{p}^* = \mathbf{p}_{j+1}$ and **stop**
12. end(**if**)
13. Set $\beta_{j+1} = \dfrac{\mathbf{r}_{j+1}^T \mathbf{r}_{j+1}}{\mathbf{r}_j^T \mathbf{r}_j}$
14. Set $\mathbf{d}_{j+1} = -\mathbf{r}_{j+1} + \beta_{j+1}\mathbf{d}_j$
15. end(**for**)
16. **return** $\mathbf{p}^*$

The first **if** condition at Line 5 terminates the algorithm if the norm of $\mathbf{p}_{j+1}$ exceeds the trust region radius $\Delta$. In this case, the solution $\mathbf{p}^*$ is located by intersection the current search direction and the trust region boundary. The scalar $\epsilon$ used in the second **if** condition at Line 10 is a user defined quantity. Setting the initial iterate $\mathbf{p}_0 = 0$ results in a very useful property, which is,

$$0 = \|\mathbf{p}_0\|_2 < \ldots \|\mathbf{p}_j\|_2 < \|\mathbf{p}_{j+1}\|_2 < \ldots \|\mathbf{p}\|_2 \leq \Delta. \tag{6.28}$$

It means that with every iteration, the algorithm increases the norm of the step until it reaches the boundary of the trust region. Any further increase will result in a step which lies outside the trust region radius and will be rejected at once (see Theorem 4.2 [NW99]).

Figure 9 depicts behaviour of the trust region Newton-CG method for minimization of the Rosenbrock function. Green points denote the iterates $\mathbf{x}_0, \ldots, \mathbf{x}_{k-1}, \mathbf{x}_k$ and red point denotes the expected next point $\mathbf{x}_{k+1}$. The black circle denotes the trust region with radius $\Delta_k$. In Figure 9a, value of the ratio $\rho_k$ is $-0.09592$ which means that $F(\mathbf{x}_{k+1}) > F(\mathbf{x}_k)$, the Newton step is rejected and the algorithm makes no progress, that is, $\mathbf{x}_{k+1} = \mathbf{x}_k$. The radius of the trust region is reduced for the next iteration as shown in Figure 9b. Now we have $\rho_k = 0.81334$ and $\|\mathbf{p}_k\| = \Delta_k$ which allow the algorithm to increase the trust region radius for the next iteration and move to next iterate.




(a) (b) (c) (d) (e) (f)

Figure 9: Minimization of the Rosenbrock function $f(x, y) = 100(y - x^2)^2 + (1 - x)^2$ using the trust region Newton-CG method. We used $\mathbf{x}_0 = (-1.5\, 2.5)$, $\bar{\Delta} = 2$, $\Delta_0 = 1$, and $\eta = 0.25$.

# 7 Numerical Experiments and Results

In this section, we use the Newton based optimization methods of Section 6 to show an improvement over the currently employed non-linear conjugate gradient method for optimization of the objective function in noise-contrastive estimation. Unnormalized Gaussian and unnormalized independent component analysis (ICA) models are used as sample data models to characterize the behaviour of the Newton-CG algorithms by comparing their computational performances. We also show that these methods perform exceptionally well in combination with the sample average approximation framework (SAA). At the end, we compare our results with the ones obtained using the non-linear conjugate gradient method.

## 7.1 Unnormalized Gaussian Model

Consider an i.i.d. sample $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{T_d})$; $\mathbf{x} \in \mathbb{R}^n$ following a zero mean Gaussian distribution having an unknown covariance matrix $\boldsymbol{\Sigma}$. We want to estimate the parameters of the unnormalized Gaussian model given as

$$p_m(\mathbf{x}; \boldsymbol{\Lambda}) \propto \exp\left( -\frac{1}{2}\mathbf{x}^T \boldsymbol{\Lambda}\, \mathbf{x} \right), \tag{7.1}$$

where $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is the inverse covariance matrix also known as the precision matrix. We assume that the model cannot be normalized in a closed form, the normalizing constant $c$ is taken as an additional unknown parameter of the model in noise-contrastive estimation. The model that we now estimate is

$$\ln p_m(\mathbf{x}; \boldsymbol{\Lambda}, c) = -\frac{1}{2}\mathbf{x}^T \boldsymbol{\Lambda}\, \mathbf{x} + c. \tag{7.2}$$

The true log-pdf of $\mathbf{x}$ is

$$\ln f_d(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T \boldsymbol{\Lambda}^*\, \mathbf{x} + c^*, \tag{7.3}$$

where the value of the normalizing constant $c^*$ which normalizes $f_d$ to integrate to 1 is given as

$$c^* = -\frac{1}{2}\ln|\det \boldsymbol{\Lambda}^*| - \frac{n}{2}\ln(2\pi). \tag{7.4}$$

For the reference dataset, an i.i.d sample $\mathcal{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_{T_n})$; $\mathbf{y} \in \mathbb{R}^n$ is generated which follows a standard Gaussian distribution. The log-pdf of $\mathbf{y}$ is

$$\ln f_n(\mathbf{y}) = -\frac{1}{2}\mathbf{y}^T\mathbf{y} - \frac{n}{2}\ln(2\pi). \tag{7.5}$$

We merge the two datasets $\mathcal{U} = \mathcal{X} \bigcup \mathcal{Y}$ and construct a label vector $\mathbf{C}$ such that $C_i = 1$ if $\mathbf{u}_i \in \mathcal{X}$ and $C_i = 0$ if $\mathbf{u}_i \in \mathcal{Y}$. The parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$ is constructed such that it contains the upper-triangular entries of the precision matrix, for example,

$$\begin{pmatrix} \Lambda_{1,1}^{(1)} & \Lambda_{1,2}^{(2)} & \cdots & \Lambda_{1,n}^{(d-n-1)} \\ \times & \Lambda_{2,2}^{(3)} & \cdots & \Lambda_{2,n}^{(d-n-2)} \\ \times & \times & \cdots & \Lambda_{3,n}^{(d-n-3)} \\ \vdots & \vdots & \vdots & \vdots \\ \times & \times & \cdots & \Lambda_{n,n}^{(d-1)} \end{pmatrix},$$

and the normalization constant $c$. Therefore, $d = \dfrac{n(n+1)}{2} + 1$ such that

$$\theta_k = \begin{cases} \Lambda_{ij}^{(k)} & \text{if } k \leq d-1 \text{ where } k = \dfrac{j(j-1)}{2} + i, \\ c & \text{if } k = d. \end{cases} \tag{7.6}$$

The log-ratio between the data and the noise distributions has the form

$$\begin{aligned} G\big(\mathbf{u};\boldsymbol{\theta}\big) &= \ln p_m\big(\mathbf{u};\boldsymbol{\Lambda},c\big) - \ln f_n\big(\mathbf{u}\big), \\ &= -\sum_{i<j} \Lambda_{ij} u_i u_j - \frac{1}{2}\sum_i \lambda_{ii} u_i^2 + c - \ln f_n(\mathbf{u}). \end{aligned} \tag{7.7}$$

The entries of the gradient vector $\nabla_{\boldsymbol{\theta}} G(\mathbf{u};\boldsymbol{\theta})$ are

$$\frac{\partial G\big(\mathbf{u};\boldsymbol{\theta}\big)}{\partial \theta_k} = \frac{\partial G(\mathbf{u};\boldsymbol{\theta})}{\partial \Lambda_{ij}^{(k)}} = \begin{cases} -u_i u_j & \text{if } i \neq j \text{ and } k \leq d-1, \\ -\dfrac{1}{2}\, u_i^2 & \text{if } i = j \text{ and } k \leq d-1. \end{cases} \tag{7.8a}$$

$$\frac{\partial G\big(\mathbf{u};\boldsymbol{\theta}\big)}{\partial \theta_d} = \frac{\partial G\big(\mathbf{u};\boldsymbol{\theta}\big)}{\partial c} = 1. \tag{7.8b}$$

Here, all the second-order partial derivatives $\dfrac{\partial^2 G\big(\mathbf{u};\boldsymbol{\theta}\big)}{\partial \theta_i \partial \theta_j} = 0$. Thus, the exact Hessian of $J_T$ is same as its Gauss-Newton approximation for the unnormalized Gaussian models.

## 7.2 Unnormalized Independent Component Analysis (ICA) Model

Consider an ICA model [HKO01] given as

$$\mathbf{x} = \mathbf{A}\mathbf{s}, \tag{7.9}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a vector of observed random variables, the sources $\mathbf{s} \in \mathbb{R}^n$ are identically distributed and independent from each other known as the independent components and $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the mixing matrix. Here, we consider Laplacian sources of unit variance and and zero mean. We want to estimate the model

$$\ln p_m\big(\mathbf{x}; \mathbf{B}, c\big) = -\sqrt{2} \sum_{i=1}^{n} |\, \mathbf{b}_i\mathbf{x}| + c. \tag{7.10}$$

where $\mathbf{B} = \mathbf{A}^{-1}$ and $\mathbf{b}_i$ denotes its $i^{th}$ row. The value of the normalizing constant which normalizes the true pdf of $\mathbf{x}$ to integrate to one is given as

$$c^* = \ln |\det \mathbf{B}^*| - \frac{n}{2} \ln 2. \tag{7.11}$$

For the reference dataset an i.i.d sample $\mathcal{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_{T_n})$; $\mathbf{y} \in \mathbb{R}^n$ is generated following a Gaussian distribution with a covariance matrix equal to the covariance matrix of $\mathcal{X}$. The log-pdf of $\mathbf{y}$ is

$$\ln f_n(\mathbf{y}) = -\frac{1}{2}\, \mathbf{y}^T \mathbf{\Sigma}^{-1} \mathbf{y} - \frac{1}{2} \ln |\det \mathbf{\Sigma}| - \frac{n}{2} \ln(2\pi). \tag{7.12}$$

We merge the two datasets $\mathcal{U} = \mathcal{X} \bigcup \mathcal{Y}$ and construct a label vector $\mathbf{C}$ such that $C_i = 1$ if $\mathbf{u}_i \in X$ and $C_i = 0$ if $\mathbf{u}_i \in Y$. The parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$; $d = n^2 + 1$, contains the rows of the matrix $\mathbf{B}$ and the normalization constant $c$ such that

$$\theta_k = \begin{cases} b_{ij} & \text{where } k = j + n(i+1) \text{ and } k \leq d-1, \\ c & \text{if } k = d. \end{cases} \tag{7.13}$$

The log-ratio between the data and the noise distributions has the form

$$\begin{aligned} G\big(\mathbf{u}; \boldsymbol{\theta}\big) &= \ln p_m\big(\mathbf{u}; \mathbf{B}, c\big) - \ln f_n\big(\mathbf{u}\big), \\ &= -\sqrt{2} \sum_{i=1}^{n} |\, \mathbf{b}_i\mathbf{u}| + c - \ln f_n\big(\mathbf{u}\big). \end{aligned} \tag{7.14}$$

The entries of the gradient vector $\nabla_{\boldsymbol{\theta}} G(\mathbf{u}; \boldsymbol{\theta})$ are

$$\frac{\partial G(\mathbf{u}, \boldsymbol{\theta})}{\partial \theta_k} = \frac{\partial G(\mathbf{u}, \boldsymbol{\theta})}{\partial b_{ij}} = -\sqrt{2}\ \frac{\mathbf{b}_i\mathbf{u}\,.\, u_j}{|\, \mathbf{b}_i\mathbf{u}|}; \quad \text{if } k \leq d-1, \tag{7.15a}$$

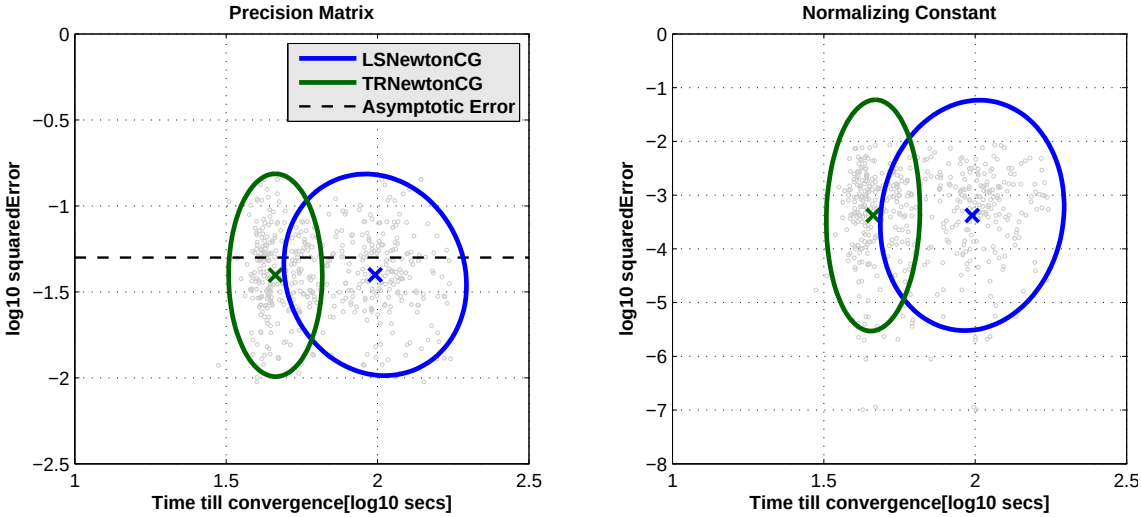
$$\frac{\partial G(\mathbf{u}, \boldsymbol{\theta})}{\partial \theta_d} = \frac{\partial G(\mathbf{u}, \boldsymbol{\theta})}{\partial c} = 1. \tag{7.15b}$$

## 7.3 Numerical Experiments: Newton-CG Methods

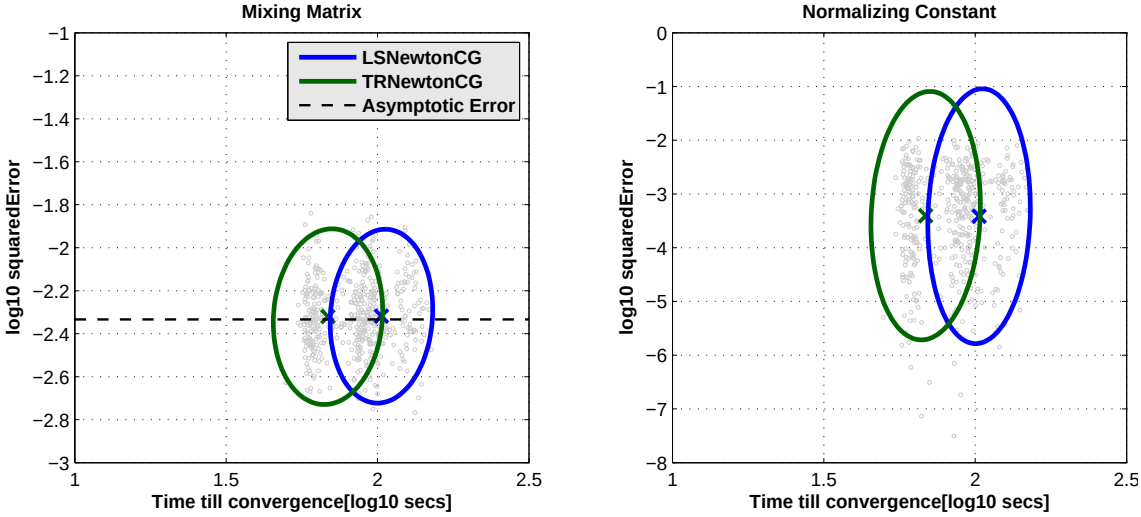
We use noise-contrastive estimation discussed in Section 2.2.2 for the estimation of the parameters of the unnormalized Gaussian model of Section 7.1 and the unnormalized ICA model of Section 7.2. The objective function obtained is optimized using both the line search Newton-CG and the trust region Newton-CG methods presented in Section 6. In order to avoid directions of negative curvature and to ensure that the resulting search direction is always a descent direction we use the Gauss-Newton approximation to the Hessian of $J_T$ as discussed in Section 6.2. We divide our experiments into three different sets. The first set of experiments compares the computational performances of the Newton-CG algorithms. We use whole sets of input samples (data and noise) for the computation of the gradient and the Hessian. In the second set of experiments, random subsets of input samples are generated during each iteration of the Newton-CG algorithms to compute the Hessian. By doing so, we are able to reduce the computation times of the Newton-CG algorithms without compromising the accuracy of the estimates. Finally, we compare our results with the ones obtained when the objective function in noise-contrastive estimation is optimized using the non-linear conjugate gradient algorithm of Rasmussen [Ras06].

### 7.3.1 Comparison of the Computational Performances of the Newton-CG Methods

Given an unnormalized model, we generated 250 random problems with $T_d = 25,000$ and $n = 10$. It has already been shown in Section 4.1 that optimizing $J_T$ for increasingly larger values of $\nu$ provides more and more accurate estimates. This does not require any further exploration and hence, we use a fixed value $\nu = 1$ for all our experiments. We ran the algorithms until no further progress, that is, decrease in the function values, was possible. We denote the estimates generated by the line search and the trust region Newton-CG algorithms by $\widehat{\boldsymbol{\theta}}_{ls}$ and $\widehat{\boldsymbol{\theta}}_{tr}$ respectively. Upon termination, the squared errors between the estimated parameters ($\widehat{\boldsymbol{\theta}}_{ls}$, $\widehat{\boldsymbol{\theta}}_{tr}$) and the true parameters ($\boldsymbol{\theta}^*$) were noted along with the total time taken by the Newton-CG algorithms to converge. Thus, every experiment produces two result points, that is, 500 result points in total. Figure 10a shows such 500 results points corresponding to the 250 random problem following the unnormalized Gaussian distribution. Estimating the precision matrix and the normalizing constant




(a) Estimation of the unnormalized Gaussian model of Section 7.1.




(b) Estimation of the unnormalized ICA model of Section 7.2.

Figure 10: Comparison of the computational performances of the Newton-CG algorithms of Sections 6.4 and 6.5. In both (a) and (b) we used $T_d = 25,000$, $n = 10$ and $\nu = 1$. The grey dots represent the result points generated by the two Newton-CG algorithms. The ellipses were obtained by fitting a Gaussian to the distribution of the result points, each one contains 90% of the results points. The blue ellipses enclose the result points generated by the line search Newton-CG method and the green ellipses contain the result points generated by the trust region Newton-CG method. The asterisks mark their centres. The black dashed line correspond to the asymptotic mean squared error of the estimates. Both the Newton-CG algorithms produce estimates of equivalent accuracy. However, the trust region strategy performs better than the line search strategy.

means estimating 56 parameters. The x-coordinate represents the time till the algorithm converges and the y-coordinate represents the estimation error at convergence. The blue ellipse contains 90% of the 250 result points generated by the line search Newton-CG method. Similarly, the green ellipse contains 90% of the 250 result points generated by the trust region Newton-CG method. The asterisks mark their centres. The black dashed line correspond to the asymptotic mean squared error of the estimators. We use it as a baseline to check the efficiency of the Newton-CG algorithms. Similar explanations follow for Figure 10b, it shows 500 results points representing 250 random problem following the unnormalized ICA distribution. Estimating the mixing matrix and the normalizing constant means estimating 101 parameters.

In Figures 10a and 10b, it is shown that both the Newton-CG algorithms produce estimates of equivalent accuracy. However, the trust region strategy computationally outperforms the line search strategy. The trust region methods calculate the search direction and the step size simultaneously using the Steihaug's algorithm. The trust region's radius for the next iteration is then adjusted in a linear amount of time requiring no additional functional or gradient evaluations. In contrast, the line search methods first invoke the conjugate gradient method to compute the search direction and then an extensive inexact line search is performed to obtain an optimal value for the step size. The inexact line search procedure requires several function and gradient evaluations during extrapolation and interpolation of candidate values for the step size. Henceforth, this additional time spent during each of the iterations leads to a slower convergence rate.

### 7.3.2 Reducing Computation Time in the Optimization

We discussed in Section 4.2 that in order to obtain better convergence results, Rasmussen's algorithm was modified to stochastically approximate the gradient of $J_T$ instead of using the exact gradient. However, when random subsets of input samples are used in optimization, the computational performance of the algorithm improves but the accuracy of the estimates goes down as well. This is because the non-linear conjugate gradient method fails to overcome the errors induced by the noisy gradients which are perpetrated through all the iterations and thus leads to less accurate estimates. Now every iteration of the Newton-CG methods is computationally intensive (of the order of $\mathcal{O}(n^3)$) as it requires computation of the Hessian

and its inverse. We use the sample average approximation in order to improve the computational performance of the Newton based optimization methods for noise-contrastive estimation. Here, the gradient is computed using whole sets of input samples whereas the Hessian is computed using random subsets of input samples. It has been shown that the sample average approximation is highly efficient in getting rid of the bottleneck present in the Newton-CG methods which is the computation of the Hessian as it is less prone to errors than the gradient [BCNN11]. Consider the $k^{th}$ iteration of a Newton-CG algorithm, the resulting search direction $\mathbf{p}_k$ is of the form

$$\mathbf{p}_k = -\left[B_{S_k}(\boldsymbol{\theta}_k)\right]^{-1} \nabla J_T(\boldsymbol{\theta}_k), \tag{7.16}$$

where $B_{S_k}(\boldsymbol{\theta}_k)$ is the Gauss-Newton approximation to the Hessian of $J_T$ computed using the random subset $S_k$ and $\nabla J_T(\boldsymbol{\theta}_k)$ is the exact gradient. In order for $\mathbf{p}_k$ to be a descent direction it must satisfy the following condition:

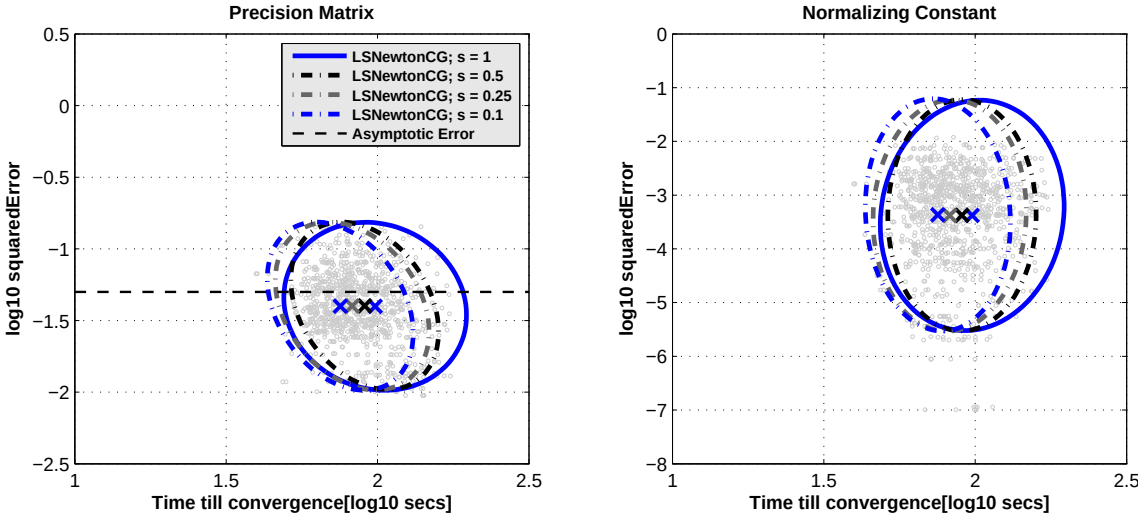
$$\nabla J_T(\boldsymbol{\theta}_k)^T \mathbf{p}_k = -\nabla J_T(\boldsymbol{\theta}_k)^T \left[B_{S_k}(\boldsymbol{\theta}_k)\right]^{-1} \nabla J_T(\boldsymbol{\theta}_k) < 0. \tag{7.17}$$

From the above equation we conclude that the Newton-CG methods will generate reliable search directions as long as the approximation matrix $B_{S_k}(\boldsymbol{\theta}_k)$ is positive definite. Now suppose that we also use the same random subset $S_k$ to compute the gradient, that is, $\nabla \widetilde{J}_{S_k}(\boldsymbol{\theta}_k)$ given in (4.4). In this situation, Equation (7.17) takes the form

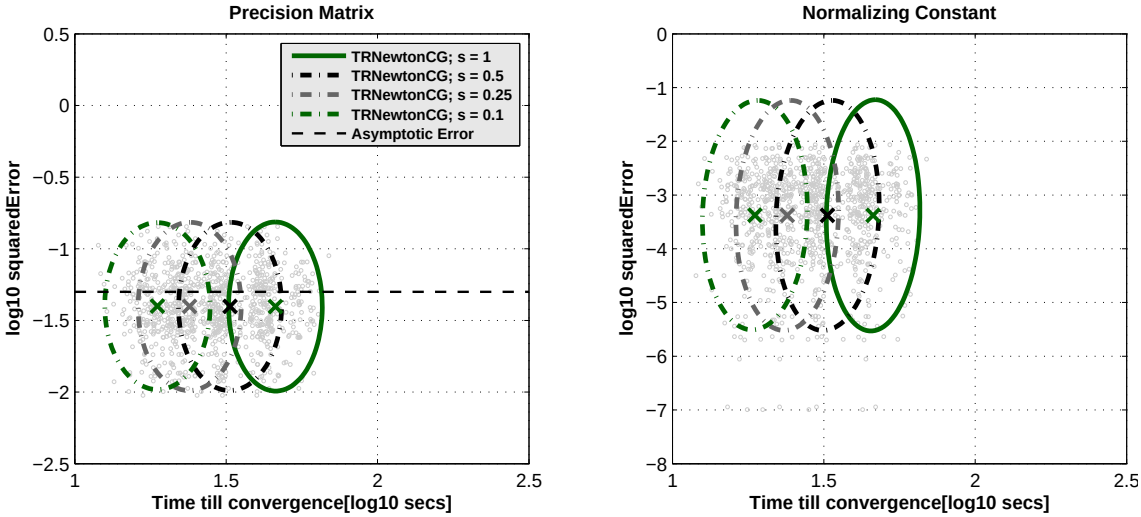
$$\nabla J_T(\boldsymbol{\theta}_k)^T \mathbf{p}_k = -\nabla J_T(\boldsymbol{\theta}_k)^T \left[B_{S_k}(\boldsymbol{\theta}_k)\right]^{-1} \nabla \widetilde{J}_{S_k}(\boldsymbol{\theta}_k) < 0. \tag{7.18}$$

We know that the matrix $B_{S_k}(\boldsymbol{\theta}_k)$ is always at least positive semi-definite however, the two non-identical vectors $\nabla J_T(\boldsymbol{\theta}_k)$ and $\nabla \widetilde{J}_{S_k}(\boldsymbol{\theta}_k)$ may cause the product $\nabla J_T(\boldsymbol{\theta}_k)^T \mathbf{p}_k$ to be non-negative. As a result, $\mathbf{p}_k$ will no longer be a descent direction. That is why, in our experiments, we computed the gradient using all input samples whereas the approximation Hessian matrix was computed using random subsets of input samples.

As shown in Section 4.1 that the best results are obtained when the subsets used to compute the gradients are switched frequently however, this increases the computation time of the non-linear conjugate gradient algorithm. Following a similar strategy we decided to generate random subsets of input samples during each iteration and monitor the behaviour of the Newton-CG algorithms. We refer to the

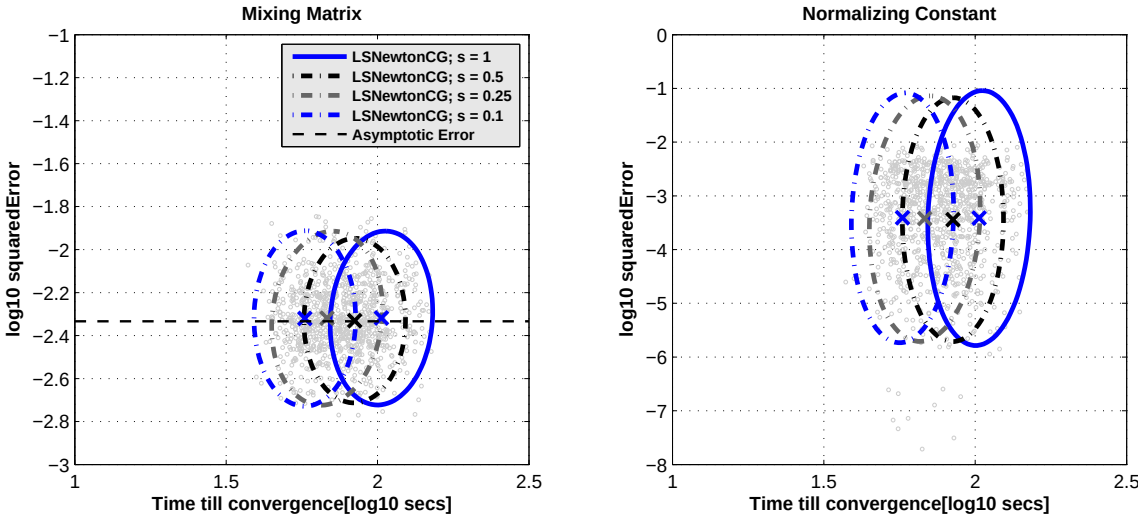



(a) Estimation of the unnormalized Gaussian model of Section 7.1. Optimization is performed using the line search Newton-CG algorithm of Section 6.4.

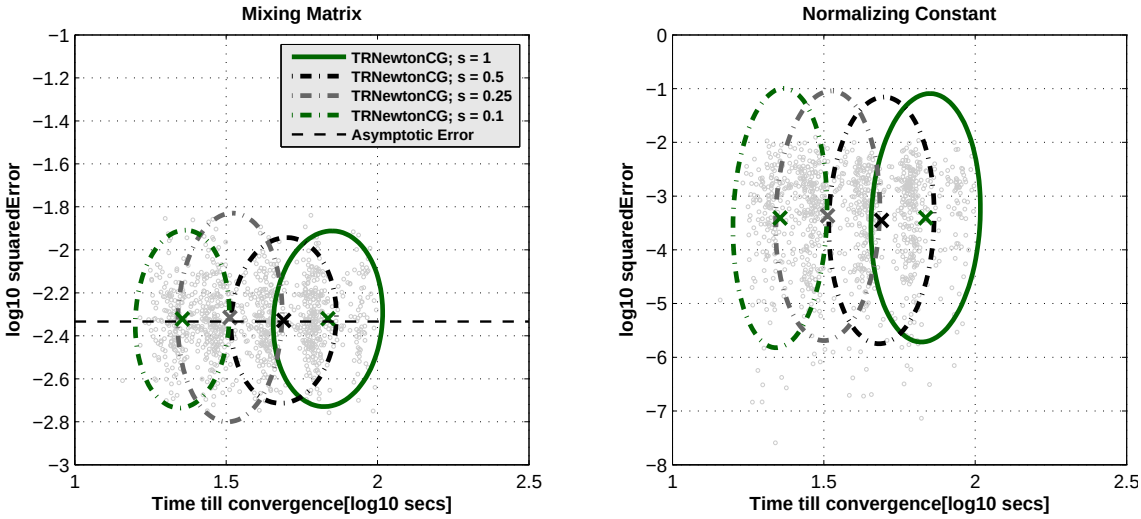



(b) Estimation of the unnormalized Gaussian model of Section 7.1. Optimization is performed using the trust region Newton-CG algorithm of Section 6.5.

Figure 11: Illustration of the sample average approximation (SAA) framework. In both (a) and (b), the four ellipses represent the outcomes for four different values of $s \in \{0.1, 0.25, 0.5.1\}$. Initially, we used $T_d = 25,000$, $n = 10$ and $\nu = 1$ hence, the updated data counts are $\widetilde{T}_d = T_d\, s$ and $\widetilde{T}_n = \nu \widetilde{T}_d$. The ellipses were obtained by fitting a Gaussian to the distribution of the result points, each one contains 90% of the results points. In both (a) and (b), the dashed black ellipse represent the results points for $s = 0.5$, that is, 50% of input samples were used to compute the Gauss-Newton approximation to the Hessian. The dashed grey ellipse represent the results points for $s = 0.25$. In (a), the blue solid and dashed ellipses correspond to the maximum and minimum values of $s$ used, that is, $s = 1$ (whole sets of input samples) and $s = 0.1$ (only 10% of input samples). Same goes for the green solid and dashed ellipses in (b). The asterisks mark the centres of the ellipses. The black dashed line correspond to the asymptotic mean squared error of the estimates.




(a) Estimation of the unnormalized ICA model of Section 7.2. Optimization is performed using the line search Newton-CG algorithm of Section 6.4.




(b) Estimation of the unnormalized ICA model of Section 7.2. Optimization is performed using the trust region Newton-CG algorithm of Section 6.5.

Figure 12: Illustration of the sample average approximation (SAA) framework. Visualized as in Figure 11.

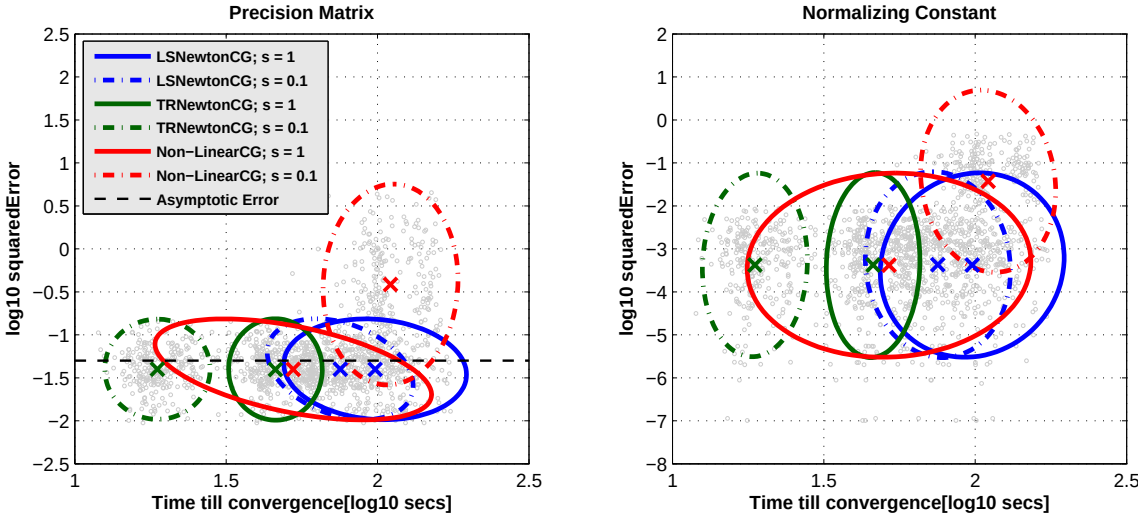
Newton-CG methods as the sub-sampled Newton-CG methods to indicate the use of sample average approximation. We need to be careful in selecting an optimal value for a subset size. The subset should not be too large such that it overcomes the cost saved by the main algorithm. On the other hand, it should not be too small so that it fails to provide any valuable curvature information. This requires further investigation on our part. Given a scalar, say $s \in \{0.1, 0.25, 0.5, 1\}$, we select random subsets of size $s$ from the datasets $\mathcal{X}$ and $\mathcal{Y}$ to compute Gauss-Newton approximation matrix. This means that the updated data counts are $\widetilde{T}_d = T_d\, s$ and $\widetilde{T}_n = \nu \widetilde{T}_d$.

Figures 11 and 12 show the results generated by this exercise. We observe that both the algorithms perform best when only 10% of input samples are used to compute the Gauss-Newton approximation to the Hessian. Unlike, the non-linear conjugate gradient algorithm which generates less precise estimates when random subsets are used, the sub-sampled Newton-CG algorithms do not compromise the accuracy of the estimates. This is a considerable improvement over the currently used optimization strategy discussed in Section 4.1.

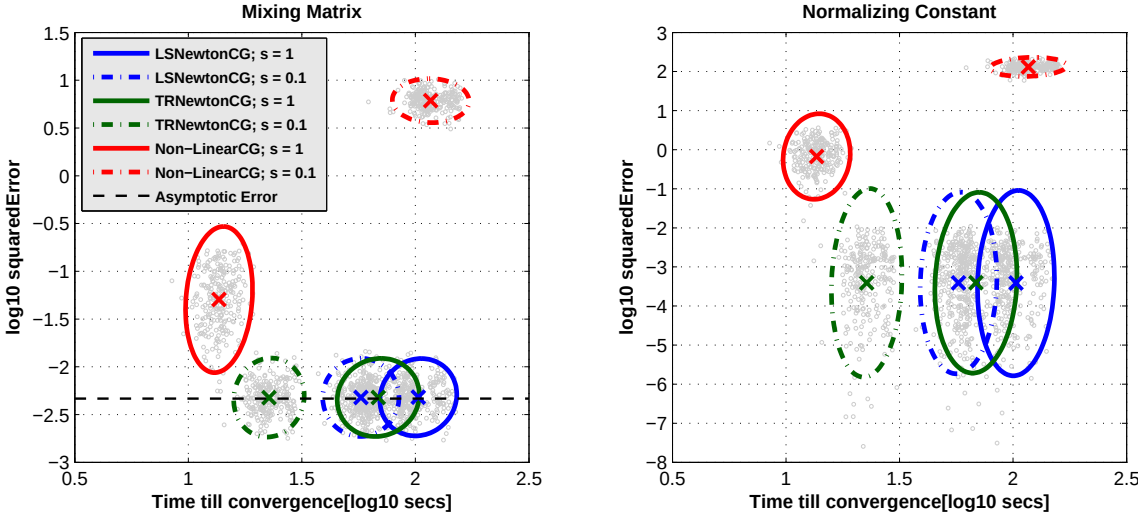
## 7.4 Results: Comparison with the Non-Linear CG Method

In the previous section, we have shown that by using random subsets of input samples to approximate the Hessian of $J_T$ we can improve the computational performances of the Newton-CG algorithms without compromising the accuracy of the estimates. Next step is to investigate how well the Newton-CG and the sub-sampled Newton-CG methods perform in contrast to the non-linear conjugate gradient method. We also compare our results with the ones obtained using the sub-sampled non-linear conjugate gradient method.

We again use the problems generated in Section 7.3 with $T_d = 25,000$, $n = 10$ and $\nu = 1$. The Newton-CG and the non-linear conjugate gradient methods use whole sets of input samples, that is, $s = 1$ during the course of optimization. The sub-sampled versions of the three methods however, generate random subsets of input samples of size $s = 0.1$ (10% of input samples) after every single update of the parameters. The Newton-CG methods use the subsets to compute the Gauss-Newton approximation matrix whereas, the non-linear conjugate gradient method uses them




(a) Estimation of the unnormalized Gaussian model of Section 7.1.




(b) Estimation of the unnormalized ICA model of Section 7.2.

Figure 13: Comparison of the Newton-CG methods with the non-linear conjugate gradient method. In both (a) and (b) we used $T_d = 25,000$, $n = 10$ and $\nu = 1$. The grey dots represent the results points generated by the optimization methods used. The ellipses were obtained by fitting a Gaussian to the distribution of the result points, each one contains 90% of the results points. The red solid and dashed ellipses represent the result points generated by the non-linear conjugate gradient algorithm of Rasmussen [Ras06] with $s = 1$ and $s = 0.1$ respectively. The blue solid and dashed ellipses correspond to the maximum and minimum values of $s$ used by the line search Newton-CG methods, that is, $s = 1$ (whole sets of input samples) and $s = 0.1$ (only 10% of input samples). Similarly, the green solid and dashed ellipses correspond to the trust region Newton-CG methods (with $s \in \{0.1, 1\}$). The asterisks mark the centres of the ellipses. The black dashed line correspond to the asymptotic mean squared error of the estimates.

to approximate the gradient. Thus, every experiment produces six result points, that is, 1500 result points in total. Figure 13 shows our final results, in both (a) and (b) the solid and dashed red ellipses contain 90% of the 250 result points generated by the non-linear conjugate gradient method with $s = 1$ and $s = 0.1$ respectively. The red asterisks mark their center. We summarize our observations as follows:

- As already discussed in Section 4 that the computational performance of the sub-sampled non-linear conjugate gradient algorithm declines with respect to the increasing frequency of random subsets generated. Moreover, sub-sampling also causes a decline in the quality of the estimates [GH12]. This is indicated by the dashed red ellipses in Figures 13a and 13b.

- In case of the unnormalized ICA model, the non-linear conjugate gradient algorithm converges faster than the Newton-CG algorithms but the resulting estimates are not accurate. This is shown in Figure 13b.

- The line-search Newton-CG algorithm (with $s \in \{0.1, 1\}$) is computationally more expensive than the non-linear conjugate gradient algorithm. Both of these algorithm use the same inexact line search procedure to compute the step size. In addition, the line search Newton-CG algorithm requires both the gradient and the Gauss-Newton approximation to the Hessian to make progress which increases its overall cost.

- Performance of the trust region Newton-CG algorithm (with $s = 1$) is almost similar to that of the non-linear conjugate gradient algorithm in Figure 13a. However, the sub-sampled trust region Newton-CG algorithm (with $s = 0.1$) performs best in both Figure 13a and Figure 13b as it converges in the least amount of time.

# 8 Future Work

Both the line search Newton-CG and the trust region Newton-CG methods use the conjugate gradient method to compute the search direction. There are some strategies that can be adopted in future which may improve the convergence rate of the conjugate gradient method. In this section, we provide a brief introduction to such strategies.

## 8.1 Preconditioning

In theory, the conjugate gradient method converges in at most $n$ steps if the symmetric positive definite Hessian $\nabla^2 F(\mathbf{x}_k)$ is well-conditioned. In other words, the rate of convergence is faster if the condition number of the Hessian is small, that is, $\kappa\big(\nabla^2 F(\mathbf{x}_k)\big) \approx 1$. In case of an ill-conditioned Hessian with a large condition number the conjugate gradient method can be accelerated by transforming the existing system of linear equations to an equivalent well-conditioned system using a technique known as *preconditioning* [AL86], [She94]. Thus, instead of solving $\mathbf{Ax} = \mathbf{b}$, we solve a related system $\widetilde{A}\widetilde{\mathbf{x}} = \widetilde{\mathbf{b}}$ for which $\widetilde{A}$ is chosen such that its condition number is closer to one.

We take a symmetric and positive definite matrix, say $\mathbf{M}$, such that $\mathbf{M} = \mathbf{D}^T\mathbf{D}$ and transform the search direction $\mathbf{p}$ to $\widetilde{\mathbf{p}}$ via the non-singular matrix $\mathbf{D}$, that is,

$$\widetilde{\mathbf{p}} = \mathbf{D}\mathbf{p}. \tag{8.1}$$

The quadratic model defined in Section 3.5 which is used to derive the Newton step is given as

$$F(\mathbf{x}_k + \mathbf{p}) \approx m_k(\mathbf{p}) = F(\mathbf{x}_k) + \nabla F(\mathbf{x}_k)^T\mathbf{p} + \frac{1}{2}\,\mathbf{p}^T\nabla^2 F(\mathbf{x}_k)\mathbf{p}. \tag{8.2}$$

Using (8.1), we can rewrite (8.2) to

$$\widetilde{m}_k(\widetilde{\mathbf{p}}) = F(\mathbf{x}_k) + \big(\mathbf{D}^{-T}\nabla F(\mathbf{x}_k)\big)^T\widetilde{\mathbf{p}} + \frac{1}{2}\,\widetilde{\mathbf{p}}^T\big(\mathbf{D}^{-T}\nabla^2 F(\mathbf{x}_k)\mathbf{D}^{-1}\big)\widetilde{\mathbf{p}}. \tag{8.3}$$

The Newton step is now obtained by solving the updated system of linear equations, that is,

$$\nabla^2\widetilde{F}(\mathbf{x}_k)\widetilde{\mathbf{p}} = -\nabla\widetilde{F}(\mathbf{x}_k), \tag{8.4}$$

where $\nabla\widetilde{F}(\mathbf{x}_k) = \mathbf{D}^{-T}\nabla F(\mathbf{x}_k)$ and $\nabla^2\widetilde{F}(\mathbf{x}_k) = \mathbf{D}^{-T}\nabla^2 F(\mathbf{x}_k)\mathbf{D}^{-1}$. Likewise in the trust region strategy, we find a constrained minimizer of the updated quadratic model given as

$$\widetilde{m}_k(\widetilde{\mathbf{p}}) = F(\mathbf{x}_k) + \big(\mathbf{D}^{-T}\nabla F(\mathbf{x}_k)\big)^T\widetilde{\mathbf{p}} + \frac{1}{2}\,\widetilde{\mathbf{p}}^T\big(\mathbf{D}^{-T}\nabla^2 F(\mathbf{x}_k)\mathbf{D}^{-1}\big)\widetilde{\mathbf{p}}; \qquad \text{s.t. } \|\widetilde{\mathbf{p}}\,\| \leq \Delta_k. \tag{8.5}$$

The non-singular matrix $\mathbf{D}$ should be chosen so that the eigenvalues of $\nabla^2\widetilde{F}(\mathbf{x}_k)$ have a favourable distribution compared to that of $\nabla^2 F(\mathbf{x}_k)$. Additionally, it is also required that $\kappa\big(\nabla^2\widetilde{F}(\mathbf{x}_k)\big) \ll \kappa\big(\nabla^2 F(\mathbf{x}_k)\big)$. The most important strategies to find such a $\mathbf{D}$ include *symmetric successive overrelaxation (SSOR)*, *incomplete Cholesky* and *banded preconditioners*, please see [Saa03] and [GVL96] for discussions of these techniques.

## 8.2 Hessian Free Optimization

The conjugate gradient method never explicitly needs the complete Hessian matrix. Given an $n$-dimensional vector, say $\mathbf{v}$, the requirement is only the Hessian-vector product which can easily be computed using finite differences at the cost of a single extra gradient evaluation via the relation provided by Pearlmutter [Pea94], that is,

$$\nabla^2 F(\mathbf{x}).\mathbf{v} = \lim_{r\to 0}\frac{\nabla F(\mathbf{x}+r.\mathbf{v}) - \nabla F(\mathbf{x})}{r} = \frac{\partial}{\partial r}\nabla F(\mathbf{x}+r.\mathbf{v})\bigg|_{r=0} \tag{8.6}$$

To put this into perspective, consider the first component of $\nabla^2 F(\mathbf{x}).\mathbf{v}$, as per normal matrix-vector multiplication, $\big[\nabla^2 F(\mathbf{x}).\mathbf{v}\big]_1$ is the dot product of the first row of $\nabla^2 F(\mathbf{x})$ and $\mathbf{v}$. Similarly, the $i^{th}$ component of $\nabla^2 F(\mathbf{x}).\mathbf{v}$ is given as

$$\big[\nabla^2 F(\mathbf{x}).\mathbf{v}\big]_i = \sum_{j=1}^{n}\frac{\partial^2 F(\mathbf{x})}{\partial x_i \partial x_j}\,v_j = \nabla\,\frac{\partial F(\mathbf{x})}{\partial x_i}.\mathbf{v}. \tag{8.7}$$

This is the directional derivative of $\dfrac{\partial F(\mathbf{x})}{\partial x_i}$ in the direction of $\mathbf{v}$ defined by the limit

$$\nabla_{\mathbf{v}}\frac{\partial F(\mathbf{x})}{\partial x_i} = \lim_{r\to 0}\frac{\dfrac{\partial F(\mathbf{x}+r.\mathbf{v})}{\partial x_i} - \dfrac{\partial F(\mathbf{x})}{\partial x_i}}{r}. \tag{8.8}$$

The above equation takes the vector form provided in (8.6) and hence, can be approximated using finite differences. The cost of computing a Hessian-vector product is $\mathcal{O}(n)$. In the worst case scenario, the conjugate gradient method will require $n$ of $\nabla^2 F(\mathbf{x}).\mathbf{v}$ products. However, if the Hessian is not well-conditioned then the number of iterations needed by the algorithm to converge will increase noticeably. This may overcome the cost saved by not using the Hessian after all [BCNN11].

# 9 Conclusions

The main focus of this thesis was on the use of Newton-based optimization methods for the optimization of the objective function in noise-contrastive estimation. Currently, the non-linear conjugate gradient algorithm of Rasmussen [Ras06] is being used for optimization. It has been shown that using more noise samples than data samples gives more and more accurate estimates [GH12]. However, this slows down the optimization process because more and more noise samples need to be processed. Thus, there exists a trade-off between statistical accuracy of the estimate and computational performance of the algorithm. It is possible to use only a fraction of input samples (data and noise) in order to reduce the computation time of the algorithm. It has been shown that when random subsets of input samples are used to compute the gradient, the computational performance of the algorithm improves but the accuracy of the estimates goes down as well [GH12].

Our objective for this thesis was to implement optimization methods that are fast, inexpensive and do not compromise the accuracy of the estimates. We investigated the line search Newton-CG and the trust region Newton-CG methods based on the Newton method [DS96] in order to achieve our research goals. The main reason that the non-linear conjugate method generates less precise estimates is its inability to overcome the errors induced by the noisy gradient during the course of optimization. The Newton based optimization methods also utilize the curvature information, that is, the Hessian in addition to the gradient of the objective function to produce better search directions. These methods will produce reliable search directions as long as the Hessian is positive definite. That is why, we opted to use the Gauss-Newton approximation to the Hessian which is always at least positive semi-definite [Che11]. Additionally, we were able to reduce the computation times of the Newton-CG algorithms by using only 10% of input samples to compute the Gauss-Newton approximation matrix without compromising the accuracy of the estimates. Both the Newton-CG algorithms produce estimates of equivalent accuracy. However, the trust region strategy computationally outperforms the line search strategy. The inexact line search procedure requires several function and gradient evaluations during extrapolation and interpolation of candidate values for the step size. Henceforth, this additional time spent during each of the iterations leads to a slower convergence rate.

# References


AL86 Axelsson, O. and Lindskog, G., On the rate of convergence of the preconditioned conjugate gradient method. *Numerische Mathematik*, 48,5(1986), pages 499–523.

BB88 Barzilai, J. and Borwein, J., Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1988), pages 141–148.

BCNN11 Byrd, R., Chin, G., Neveitt, W. and Nocedal, J., On the Use of Stochastic Hessian Information in Optimization Methods for Machine Learning. *SIAM Journal on Optimization*, 21,3(2011), pages 627–642.

Bis95 Bishop, C., *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., 1995.

BT99 Bertsekas, D. and Tsitsiklis, J., Gradient Convergence in Gradient Methods with Errors. *SIAM Journal on Optimization*, 10,3(1999), pages 627–642.

Cau47 Cauchy, M., Méthode générale pour la résolution des systèmes d'équations simultanées*. *Comptes Rendus Hebd. Séances Acad. Sci.*, 25(1847), pages 536–538.

CGT12 Cartis, C., Gould, N. and Toint, P., On the complexity of the steepest-descent with exact linesearches. Technical Report, 2012.

Che11 Chen, P., Hessian Matrix vs. Gauss-Newton Hessian Matrix. *SIAM Journal On Numerical Analysis*, 49,4(2011), pages 1417–1435.

DS96 Dennis, J. and Schnabel, R., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1996.

DY99 Dai, Y. and Yuan, Y., A Nonlinear Conjugate Gradient Method with a Strong Global Convergence Property. *SIAM Journal on Optimization*, 10,1(1999), pages 177–182.

Fle87 Fletcher, R., *Practical Methods of Optimization*. John Wiley & Sons, Inc., 1987.

FR64 Fletcher, R. and Reeves, C., Function minimization by conjugate gradients. *The Computer Journal*, 7,2(1964), pages 149–154.

GH12 Gutmann, M. and Hyvärinen, A., Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. *Journal of Machine Learning Research*, 13(2012), pages 307–361.

GH13 Gutmann, M. and Hyvärinen, A., Estimation of unnormalized statistical models without numerical integration. *The Sixth Workshop on Information Theoretic Methods in Science and Engineering (WITMSE2013)*, Tokyo, Japan, 2013.

GM98 Gelman, A. and Meng, X., Simulating Normalizing Constants: From Importance Sampling to Bridge Sampling to Path Sampling. *Satistical Science*, 13,2(1998), pages 168–185.

GNS08 Griva, I., Nash, S. and Sofer, A., *Linear and Nonlinear Optimization*. Society for Industrial and Applied Mathematics, 2008.

GVL96 Golub, G. and Van Loan, C., *Matrix Computations*. Johns Hopkins University Press, 1996.

Has61 Haselgrove, C., A Method for Numerical Integration. *Mathematics of Computation*, 15,76(1961), pages 323–337.

HKO01 Hyvärinen, A., Karhunen, J. and Oja, E., *Independent Component Analysis*. John Wiley & Sons, Inc., 2001.

HS52 Hestenes, M. and Stiefel, E., Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, 49,6(1952), pages 409–436.

Hyv05 Hyvärinen, A., Estimation of non-normalized statistical models using score matching. *Journal of Machine Learning Research*, 6(2005), pages 695–709.

HZ05 Hager, W. and Zhang, H., A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM Journal on Optimization*, 16,1(2005), pages 170–192.

KPH15 Kim, S., Pasupathy, R. and Henderson, S., A guide to sample-average approximation for simulation optimization. *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research  Management Science*. Springer New York, 2015, pages 207–243.

KS80 Kindermann, R. and Snell, J., *Markov Random Fields and Their Applications*. American Mathematical Society, 1980.

Lay12 Lay, D., *Linear Algebra and Its Applications: International Edition*. Pearson Education, Inc., 2012.

LS91 Liu, Y. and Storey, C., Efficient Generalized Conjugate Gradient Algorithms, Part 1: Theory. *Journal of Optimization Theory and Applications*, 69,1(1991), pages 129–137.

Myu03 Myung, I., Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47,1(2003), pages 90–100.

NW99 Nocedal, J. and Wright, S., *Numerical Optimization*. Springer-Verlag, 1999.

Pea94 Pearlmutter, B., Fast Exact Multiplication by the Hessian. *Neural Computation*, 6,1(1994), pages 147–160.

Pol69 Polak, E., R. G., Note sur la convergence de méthodes de directions conjuguées. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 3,1(1969), pages 35–43.

Ras06 Rasmussen, C., Conjugate gradient algorithm, matlab code version 2006-09-08, 2006. URL `http://learning.eng.ca.ac.uk/carl/code/minimize.m`.

RPKL07 Robins, G., Pattison, P., Kalish, Y. and Lusher, D., An introduction to exponential random graph ($p^*$) models for social networks. *Social Networks*, 29,2(2007), pages 173–191.

Saa03 Saad, Y., *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003.

She94 Shewchuk, J., An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Technical Report, 1994.

Ste86 Steihaug, T., The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20,3(1986), pages 626–637.

Val08 Valpine, P., Improved Estimation of Normalizing Constants From

Markov Chain Monte Carlo Output. *Journal of Computational and Graphical Statistics*, 17,2(2008), pages 333–351.

Wol69 Wolfe, P., Convergence Conditions for Ascent Methods. *SIAM Review*, 11,2(1969), pages 226–235.

Wol71 Wolfe, P., Convergence Conditions for Ascent Methods. II: Some Corrections. *SIAM Review*, 13,2(1971), pages 185–188.